JLC-HDBK-SRAH

March 1997

TECHNICAL REPORT

SOFTWARE REENGINEERING ASSESSMENT HANDBOOK

Version 3.0

This page intentionally left blank.

EXECUTIVE SUMMARY

Existing software is a valuable DoD asset which should be leveraged to the greatest degree possible. There is an immediate need for DoD guidance in conducting technical, economic, and management assessments to determine when software reengineering techniques are cost beneficial.

Such guidance has been developed under the auspices of the Joint Logistics Commanders, Joint Group on Systems Engineering (JLC-JGSE). This Technical Report introduces and details the new *Software Reengineering Assessment Handbook (JLC-HDBK-SRAH) Version 3.0*.

The *SRAH*, formerly titled the *Reengineering Economics Handbook (REH)*, is intended to be used as guidance for conducting an effective technical, economic, and management assessment of existing software to determine whether to maintain, reengineer, or retire that software. Reengineering strategies include: reverse engineering, restructuring, translation, data reengineering, redocumentation, forward engineering, and retargeting.

The decision to reengineer existing software versus maintain or retire that software should be based on standard criteria and a defined process to determine *if*, *when*, and *how* to reengineer. This handbook provides the documentation of a Software Reengineering Assessment (SRA) process including a cost/benefit methodology for DoD. The SRA process is applicable to all types and sizes of software and all levels of an organization, as well as non-DoD, commercial/industrial, and academic organizations, and includes three sequential processes:

1. **Reengineering Technical Assessment (RTA) Process:** The evaluation of candidate software and the selection of reengineering strategies for each candidate using weighted technical criteria and other attributes.

2. **Reengineering Economic Assessment (REA) Process**: The evaluation of the reengineering strategies and the optional selection of preferred strategies using economic indicators.

3. **Reengineering Management Decision (RMD) Process**: The evaluation and rank ordering of candidates and their reengineering strategies using a combination of technical, economic, and other considerations.

The SRA process can be applied to three situations: *Case 1* B Assessing a set of candidates within an organization to determine which has the greatest need to reengineer, and determining if, when, and how to reengineer; *Case 2* B Assessing a specific candidate within the organization; and *Case 3* B Assessing a specific reengineering strategy for a specific candidate.

This handbook is available for use by all departments and agencies of the Department of Defense (DoD). Recommendations for changes, additions, deletions, and/or pertinent data or feedback on its use which may be of use in improving this document should be addressed to the following:

Mr. Robert E. Johnson, Jr.
Single Agency Manager
Architecture and Reengineering
104 Boundary Channel Drive
Arlington, VA 22202-3700
VOICE: 703-693-2740 or DSN 223-2740
FAX: 703-693-2929 or 2888
Email: johnsonro@pentagon-sam.army.mil

Copies of the SRAH Version 3.0 can be downloaded from the USAF/STSC web site at: http://www.stsc.hill.af.mil/

Paper copies can be obtained from the DTIC at 315-334-4933.

CD copies can be obtained from DACS at 315-334-4905.

TABLE OF CONTENTS

APPENDICES

LISTS OF FIGURES AND TABLES

LIST OF FIGURES

LIST OF FIGURES (CONT.)

List of Figures (cont.)

LIST OF TABLES

1. Introduction

1.1 Scope

This *Software Reengineering Assessment Handbook (SRAH)* provides information and guidance to individuals responsible for DoD software, specifically in maintaining existing software, implementing software reuse, and incorporating Commercial Off The Shelf (COTS)/ Non-Developmental Items (NDI) into new or existing systems. It focuses on providing practical assistance to the analyst and/or engineer responsible for analyzing existing software for potential reengineering. However, the results of the processes described in this handbook are intended to be used by DoD decision makers as a basis for making informed decisions about whether to maintain, reengineer, or retire existing software. Portions of this process may not be fully applicable to every type of software and may be tailored to each user organization's needs. The following terms are used throughout this handbook:

- **Maintenance –** the continued support of existing software (i.e., status quo).
- **Reengineering –** the examination and alteration of an existing subject software component. This process encompasses a combination of subprocesses such as reverse engineering, restructuring, translating, data reengineering, redocumenting, forward engineering, and retargeting. Emphasis is placed on consolidation, migration, or reuse activities that will lead to shared integrated resources.
- **Retirement –** discontinuing use of a software component.

1.2 Applicability

This handbook applies to the analysis and planning of reengineering efforts for all DoD software. Use of this handbook should be consistent with the provisions of applicable directives. This handbook also applies to other Government and industrial software, (the same principles and processes apply). It provides a defined Software Reengineering Assessment (SRA) process for conducting an effective technical, economic, and management assessment of existing software to determine whether to maintain, reengineer, or retire that software. The technical and economic assessment sub-processes are intended for execution by individuals having a thorough understanding of the existing software, and software economics, or access to someone who has this knowledge. The management decision sub-process is intended for execution by a decision maker familiar with these areas, but not necessarily expert in any of them.

The overall SRA process can be performed at the system level, the Computer Software Configuration Item (CSCI) level for embedded/tactical systems, or the computer program level for Automated Information Systems (AIS). Its applicability ranges from small to very large systems and from a single maintainer to DoD executives responsible for software. The SRA process can be applied at the maintenance organization level, the program office level, or at the systems command level.

The SRA process is applicable to various types of project life cycle approaches including waterfall, incremental, periodic, and spiral. It will be applied more frequently for incremental, periodic, and spiral reengineering, than for waterfall, but on a smaller scale.

The SRA process is designed to be tailored to meet the needs of certain types/classes of software. For example, when tactical (hard real-time, embedded, or weapon-related software) software components are considered, the reengineering decision maker (e.g., Program Manager) may not be choosing between several software components for the best to reengineer. Rather, he/she is faced with a go/no-go choice on reengineering, or is reengineering for other than pure economic reasons and only needs to choose the best available strategy. Thus, some of the handbook processes are not applicable. Furthermore, software engineering trade-off analysis may precede the SRA to address open technical issues. Each user should tailor the SRA process to fit their organization's needs.

1.3 Motivation

Existing software is a valuable DoD asset which must be used to the greatest degree possible. Consequently, there is an immediate need for DoD guidance for conducting technical, economic, and management assessment of reengineering techniques to determine when they are cost beneficial. The *Software Reengineering Executive Strategy and Master Plan* produced at the JLC-JGSE Santa Barbara I Reengineering Workshop (reference 2.1.2e) states:

> The state of DoD software systems is approaching a national crisis. Currently 30% of the software budget is being spent on development, while the other 70% is spent on maintenance. Woefully, 50% of the maintenance resources are expended on understanding system requirements and the design and specifications of existing systems. This catastrophe has drastically affected our war fighting capability and drained our resources. Resources currently devoted to continued maintenance could be better utilized on reengineering or new development with greater return on investment. This alarming trend will be exacerbated unless there is a devoted effort to change the entire process of the system acquisition from cradle to grave. Utilizing systems software reengineering and reuse as a paradigm for capitalizing on our investment in existing and future systems shows great potential for resolving a major portion of the problem ... The vision of this Master Plan is to preserve, extend, and leverage DoD's past and present investments in systems through reengineering for the future. Bold leadership is required to achieve the following goals for the year 2000:

- For existing systems – Reengineer based on return on investment assessment
- For new systems – Develop with reengineering and/or reuse structures
- For technology – Enable working at high levels of aggregation (e.g., configurations) and abstraction (e.g., requirements and specifications)
- For infrastructure – Foster consistent policies, standards, procedures, education, tools, incentives, and business practices that integrate reengineering into the systems engineering process

Technical reasons to reengineer existing software include mission requirements, major enhancements, modifications for new applications, rehosting to a new computer, translation to a new language, etc., which eventually come down to an economics decision. This Handbook provides the required DoD guidance for conducting technical, economic, and management assessments to determine when software reengineering techniques are cost effective.

1.4 Software Reengineering Assessment (SRA) Process

The factors leading to the decision to reengineer are complex. The goal of this handbook is to present those factors in a straightforward manner. Decisions regarding reengineering are critical since they involve the allocation of an organization's resources: *money*, *time*, and *personnel*.

Figure 1-1 presents an overview of the SRA Process. Subsequent chapters discuss the individual parts of the process in greater detail. There are multiple entry and exit points in the process. Candidate software (e.g., computer programs) enters the process based on a number of factors listed in Section 4 including those candidates perceived as being a problem. Candidates specifically directed to be reengineered will not need to be screened. Similarly, candidates that were previously identified through some other process for reengineering can enter the SRA process at the beginning of the economic process bypassing the technical process. However, the user may want to subject such candidates to the entire process to determine if a different strategy is more appropriate and/or to validate the process itself.

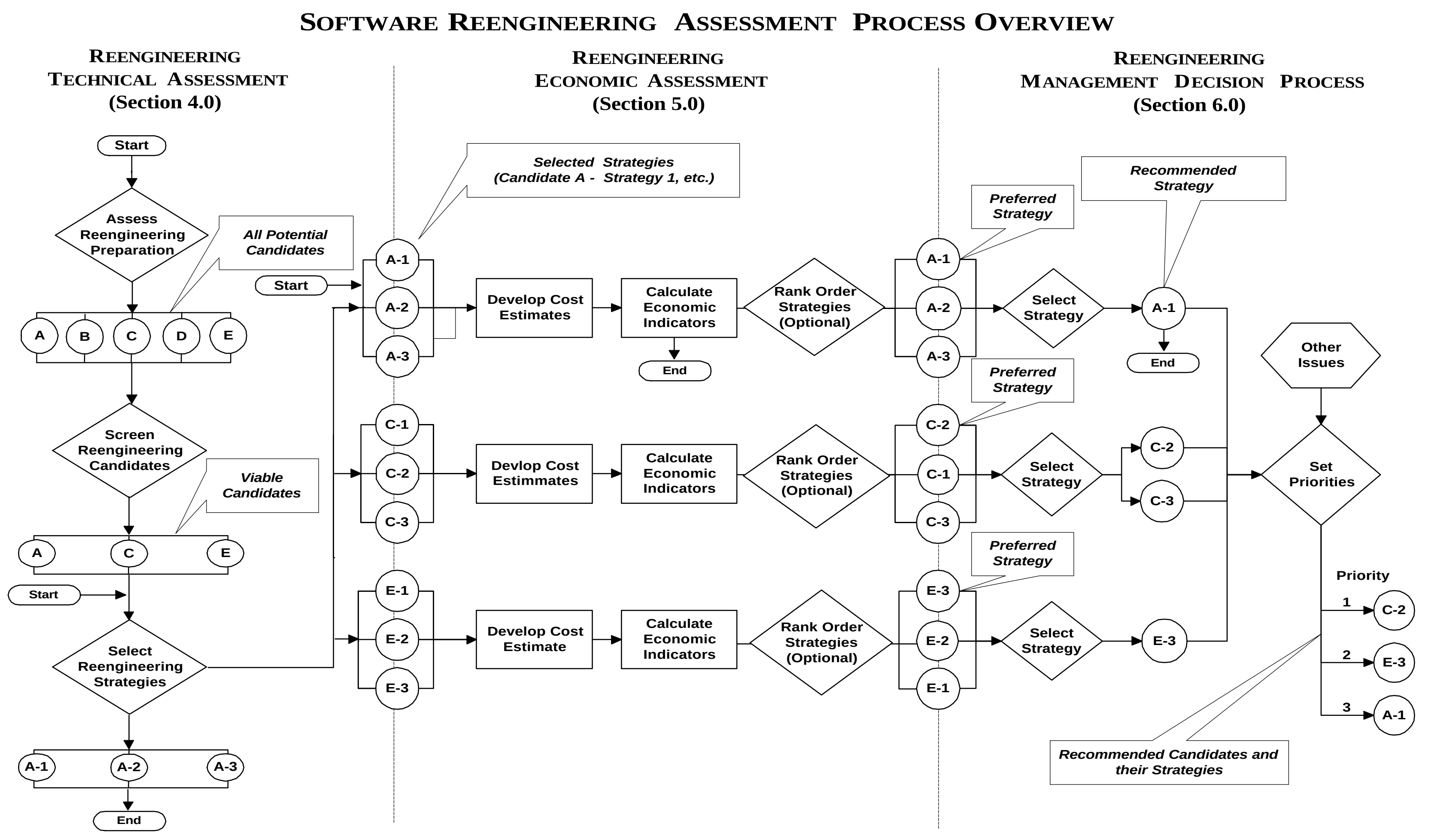

SOFTWARE REENGINEERING ASSESSMENT PROCESS OVERVIEW
REENGINEERING TECHNICAL ASSESSMENT (Section 4.0)
REENGINEERING ECONOMIC ASSESSMENT (Section 5.0)
REENGINEERING MANAGEMENT DECISION PROCESS (Section 6.0)
Start
Assess Reengineering Preparation
All Potential Candidates
A
B
C
D
E
Screen Reengineering Candidates
Viable Candidates
Start
Select Reengineering Strategies
A-1
A-2
A-3
End
Selected Strategies (Candidate A - Strategy 1, etc.)
Start
Develop Cost Estimates
Devlop Cost Estimmates
Develop Cost Estimate
Calculate Economic Indicators
End
Rank Order Strategies (Optional)
C-1
C-2
C-3
E-1
E-2
E-3
Preferred Strategy
Recommended Strategy
Select Strategy
End
Other Issues
Set Priorities
Priority
1
2
3
Recommended Candidates and their Strategies

Figure 1-1. Software Reengineering Assessment (SRA) Process Overview

As previously described, the SRA process consists of three distinct components: *technical*, *economic*, and *management*.

A. Reengineering Technical Assessment (RTA) Process

1. Assess the organization's level of preparation to reengineer (Section 4.2).

2. Identify candidate software (Section 4.3).

3. Reduce the list of candidates to determine their viability for reengineering (Section 4.4).

4. Complete the RTA Questions, Table 4-2, for each candidate. Identify the reengineering strategies based on the scores from the questionnaire (Section 4.5).

5. Consider other strategies (Section 4.6).

6. If evaluating multiple candidates, enter the RTA Questionnaire results into the Detailed Assessment Results (DAR) worksheet in Appendix C, and answer the Maintenance Environment Question Set (Table 4.1).

7. Document the technical assessment results (Section 4.13).

B. Reengineering Economic Assessment (REA) Process

8. Establish ground rules and assumptions (Section 5.4).

9. Establish cost element structure (Section 5.5).

10. Determine estimating methods and collect data (Section 5.6).

11. Develop cost estimates (Section 5.7).

11.1 Perform cost sensitivity analysis (optional, Appendix D.1).

11.2 Perform cost risk analysis (optional, Appendix D.2).

12. Perform present value analysis and calculate economic indicators (Section 5.8)

13. Select preferred strategies (optional Section 5.9).

C. Reengineering Management Decision (RMD) Process

14. Prepare management report (Section 6.2).

15. Select the reengineering projects, and establish project priorities (Section 6.3).

16. Implement and document the management decision results (Section 6.4).

1.5 Content

Sections 1 through 6 contain the technical, economic, and management decision sections and related Appendices A through D. Appendices E through J, contains information, guidance, and examples for using specific models to estimate reengineering efforts. The examples were based on a case study which left much room for individual interpretation. Consequently, each model developer assumed slightly different attributes of the case study, which reengineering strategies were more beneficial/economical, and which approach should be taken for a particular strategy.

The results of the various models should not be compared with each other because the assumptions differed and the models are based on different project databases. The purpose of providing these appendices is to show that the models do support estimating reengineering efforts. In some cases, the model developers provided significant insight and innovation based on their experiences with actual reengineering projects to date.

2. Applicable Documents

2.1 Government Documents

2.1.1 Specifications, Standards, and Handbooks

This handbook references the following Government specifications, standards, or other handbooks:

a. *MIL-STD-881B, Work Breakdown Structures for Defense Materiel Items*, 25 March 1993.

b. *MIL-HDBK-171, Work Breakdown Structure for Software Element*, July 1995.

c. *MIL-HDBK-347, Mission Critical Computer Resources Software Support*.

d. *DoD-STD-2167A, Defense System Software Development*.

e. *MIL-STD-498*.

f. *MIL-STD-499*.

2.1.2 Other Government Documents and Publications

The following Government documents and publications are also referenced in this handbook:

a. *DoD 7041.3*, 18 October 1972, and *AFR 173-15*, 4 March 1988, "Economic Analysis and Program Evaluation for Resource Management."

b. *Department of the Army, Economic Analysis Manual*, U.S. Army Cost and Economics Analysis Center, Washington, D.C., August 1992.

c. *AFSC Cost Estimating Handbook*, Air Force System Command, Andrews AFB, D.C., Undated.

d. *OMB Circular A-94*.

e. *PDSS Cost Management and Process Control: Work Breakdown Structure, Cost Structure, and Data Collection, JLC/JPCG-CRM Draft Report*, 31 March 1994, Comptek Federal Systems, Inc.

f. *Proceedings of the First Software Reengineering Workshop, JLC-JPCG-CRM*, Santa Barbara I, September 1992, Joint Logistics Commanders.

g. Sittenauer, Chris, and Michael Olsem, "Back to the Future Through Reengineering: The Santa Barbara I Reengineering Workshop," *CrossTalk*, November 1992.

h. *Reengineering Technology Report*, Volumes 1 and 2, Software Technology Support Center (STSC), August 1993.

i. *Software Estimation Technology Report*, Software Technology Support Center, (STSC), March 1993.

There are numerous other DoD documents and federal documents which may provide additional information to help facilitate application of this handbook:

1. *DoD Enterprise Model*
2. *Technical Reference Model*
3. *Technical Architectures Framework for Information Management*
4. *CIM Software Systems Reengineering Process Model*
5. *DoD 8000.1 B Defense Information Management (IM) Program*
6. *DoD 8020.1-M B Functional Process Improvement (Functional Management Process for Implementing the Information Management Program of the Department of Defense)*
7. *DoD 8320 B DoD Data Administration.*
8. *A Plan for Corporate Information Management for the Department of Defense*
9. *CIM Business Process Improvement Program Implementation Pla*n
10. *Information Systems Criteria for Applying Software Reengineering*
11. *Functional Economic Analysis (FEA)* B Brochure
12. *Functional Economic Analysis Guidebook (CIM)*
13. *Process Improvement Methodology for DoD Functional Managers*
14. *User's Manual, Function Economic Analysis Model* (2.2.a).

2.2 Non-Government Documents and Publications

The following non-Government documents and publications are referenced:

a. Chikofsky, Elliot J., and James H. Cross II, "Reverse Engineering & Design Recovery: a Taxonomy," *IEEE Software*, pp. 13-17.

b. Boehm, Barry, *Software Engineering Economics*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1981.

c. *PRICE SJ Reference Manual*, Third Edition, Martin Marietta PRICE Systems, Inc. Cherry Hill, NJ, October 1993.

d. *SEER-SEM User's Manual*, Galorath Associates, Inc., Los Angeles, CA, March 1996.

e. *Measures for Excellence: Reliable Software, on Time, within Budget*, Lawrence H. Putnam and Ware Myers, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1992.

f. *SLIM User's Manual*, Quality Software Measurement, Inc.

g. Londeix, Bernard, *Cost Estimation for Software Development*, Addison Wesley, 1987.

h. Wood, Michael, "A Reengineering Economics Model," *CrossTalk*, June/July 1992.

i. McCabe, et. al., "Structured Real-Time Analysis and Design," *IEEE COMPAC 85*, October 1985.

j. McCabe T., "A Complexity Measure," *IEEE Transactions on Software Engineering*, SE-2, 308-320, 1976.

k. *SoftCost-Ada User's Manual*, *Reference Manual*, and *Installation and Tutorial Manual*, Resource Calculations, Inc.

l. *CHECKPOINT User's Guide*, Software Productivity Research, Inc.

2.3 Reengineering Preparation and Project Planning Models

Several DoD organizations and private enterprises have defined reengineering preparation and project planning models. Two DoD agencies that have defined such models include:

- USAF Software Technology Support Center, OO-ALC/TISEC, 7278 Fourth Street, Hill AFB, Utah 84056-5205. Phone number (801) 777-8057 or DSN 458-8057.

- DISA/CIM, 701 South Courthouse Rd., Arlington, VA 22204-2199. Phone number (703) 285-6589 or DSN 356-6589.

3. Acronyms and Definitions

Analogy Estimating: An estimating method based on comparison of the software to be estimated with software that has been developed for known costs.

Assumptions: Statements or hypotheses made concerning uncertain factors and data that are required to complete the effort. They are not known facts.

Benefit: Results expected in return for costs and inputs incurred or used. A positive output of an alternative. Quantifiable benefits include cost savings, cost avoidance, and productivity improvement (personnel reductions). Non-quantifiable benefits include improved morale, quality, security, compatibility, and readiness *(USA Economics Analysis Manual)*.

Benefit Investment Ratio (BIR): The ratio of the present value of the dollar quantifiable benefits (savings and cost avoidances) divided by the present value of the investment cost (development, production, military construction, and fielding) of the alternative. It does not include benefits associated with sunk costs. *(USA Economics Analysis Manual).*

Break-even Point (BP): The point in time when the benefit in current dollars equals the investment in current dollars. It does not include sunk costs. Also, the point where the total cost of an alternative in current dollars equals the total cost of another alternative in current dollars. (*USA Economics Analysis Manual*).

Computer Software Component (CSC): A distinct part of a CSCI. CSCs may be further decomposed into other CSCs and CSUs *(DoD-STD-2167A)*.

Computer Software Configuration Item (CSCI): A configuration item for computer software *(DoD-STD-2767A).*

Computer Software Unit (CSU): An element specified in the design of a CSC that is separately testable *(DoD-STD-2167A)*.

Configuration Item (CI): An aggregation of hardware or software that satisfies an end use function and is designated by the government for separate configuration management *(MIL-STD-973).*

Constant-Dollars: All prior year, current, and future costs that reflect the level of prices of a base year. Constant dollars have the effects of inflation removed. *(USA Economic Analysis Manual)*. Constant dollars are normalized to the purchasing power of a dollar in a base year. Use of constant dollars allows consistent comparisons of costs between different years because of compensation for the effects of inflation and deflation.

Cost Avoidance: All reductions in future resource requirements, not in an approved POM or MDEP, because investment in some needed program/project will not have to be made *(USA Economics Analysis Manual)* (See Cost Savings).

Cost Element Structure (CES): The software product-oriented tree structure to which all life cycle cost accounts may be assigned; similar to a work breakdown structure (WBS).

Cost Savings: A cost reduction that is made in a specific POM or MDEP from implementing a specific alternative that does not degrade current capability, instead of the present system. Savings are a quantifiable benefit. (*USA Economics Analysis Manual*). USA Economics Analysis policy defines cost savings as a benefit occurring only during a POM or MDEP period, and cost avoidance as a benefit occurring both during and following a POM or MDEP period but not part of either. Cost avoidance and cost savings are mutually exclusive.

Current Dollars: Dollars that reflect the purchasing power of the dollar in the year the cost or savings is to be realized or incurred. That is, current dollars reflect the effects of inflation. Prior year costs stated in current dollars reflect actual present-day costs. Future costs or savings stated in current year dollars are the projected values which will be paid out or saved in future years. *(USA Economics Analysis Manual)* Prior, current, and future dollars that have been adjusted to reflect the reduced purchasing power of dollars received in the future. This reflects the fact that dollars received or spent in the future are worth less than dollars today; therefore, more dollars will be needed in the future to purchase the same item. Current Dollars are calculated from specific Constant Dollars using weighted inflation rates.

Cyclomatic Complexity: A measurement of the number of linearly independent paths through a program: the number of control verbs plus one. (See "A Complexity Measure", *IEEE Transactions on Software Engineering*, SE-2, 308-320, 1976.

Data Reengineering: Transformation of data from one format to another format (e.g., going from flat-file to relational database format) (Santa Barbara I).

Data Standardization: The "process of reviewing and documenting the names, meanings, and characteristics of data elements so that all users of the data have a common, shared understanding of it. Data standardization is a critical part of the DoD Data Administration Program, managed under *DoD Directive 8320.1*. Data administration is the function that manages the definition and organization of the Department's data" [Memorandum data October 13, 1993 subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement," page 3].

Discounting: A technique for converting various annual cash flows over a period of time to equivalent amounts at a common point in time, considering the time value of money, in order to facilitate comparison (*USA Economic Analysis Manual*).

Discounted Dollars: Future costs that have been discounted to reflect the time value of money.

Discount Rate: The interest rate used to discount or calculate future costs and benefits so as to arrive at their present values *(DoD 7041.3)*. This term is also known as the opportunity cost of capital investment (*USA Economic Analysis Manual)*.

Economic Assessment: A process for evaluating the relative cost differences between candidate strategies for reengineering software.

Economic Indicators: Metrics that can be used to appropriately compare various alternative strategies being considered. These indicators include Present Value of Total Cost PV(TC), Present Value of Total Benefits PV(TB), Net Present Value (NPV), Benefit Investment Ratio (BIR), Break-even Point (BP), and Rate of Return (ROR). NPV is preferred.

Essential Complexity: A measurement of the level of "structuredness" of a program. (See "Structured Real-Time Analysis and Design" by McCabe et. al., *IEEE COMPSAC-85*, October 1985.)

Existing Software: Existing source code, assembly code, documentation, or code implemented in firmware being evaluated for potential reengineering.

Existing System: The existing hardware, software, and/or firmware that is being evaluated for potential reengineering.

Forward Engineering: The set of engineering activities that consume the products and artifacts derived from existing software and new requirements to produce a new target system (Santa Barbara I).

> Note: Notice the difference between software engineering and forward engineering. An issue within the software reengineering community is whether the term "forward engineering" is needed since it implies the normal development life cycle sequence of events. If this were true, then forward engineering and software engineering can be considered identical terms. But forward engineering can be defined as using the output of reverse engineering. This implies some reengineering must have occurred prior to the forward engineering activity. For example, the most common forward engineering activity involves the generation of source code from design information which was captured by a previous reverse engineering activity.

Function Points: The unit of measure used in Function Point Analysis (FPA). It is a technique for measuring the "size" of a software project or system. It is based on the functionality of the software expressed in terms of five components: (1) internal logical files, (2) external interface files, (3) external inputs, (4) external outputs, (5) external inquiries. Each component is rated according to a qualifying criteria provided in the *International Function Point Users Group (IFPUG) Counting Practices Manual* (*STSC Software Estimation Technology Report*, "Appendix G," March 1993).

Ground Rule: A basic rule or fact that cannot be altered and which constrains an analyst from considering an alternative course of action.

Initial Operating Capability (IOC): The date at which operational acceptance testing is complete and when support begins. The first capability attainment to imply effectively a weapon system, an item of equipment, or system of approved characteristics, and which is manned or operated by a trained, equipped, and supported military unit (Defense Systems Management College).

Investment Cost: Includes Research and Development (R&D) and Production and Deployment costs of the system (*USA Economic Analysis Manual*). Cost elements include software development, site preparation, training, development tools, and hardware. Investment does not include the Operations and Support (O&S) cost to maintain either the existing system or the reengineered system.

JLC-JGSE: Joint Logistics Commanders, Joint Group on Systems Engineering.

Legacy System: One of a set of "other AISs...that duplicate the support services provided by the migration system." [Legacy systems] are terminated, so that all future AIS development and modernization can be applied to the migration system (Deputy Secretary of Defense memorandum dated October 13, 1993; subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement," page 4).

> Note: The term "legacy software" arises from the fact that these systems are frequently unstructured, undocumented, and are the legacy of earlier, less mature days of software development.

Level of Effort (LOE): An estimating method based on the number of labor hours and a fixed labor rate.

Life Cycle Cost Estimate (LCCE): Includes all costs incurred during the total life of a project, from initiation through termination. The LCCE includes the costs for Research and Development, Production and Deployment, and Operations and Support (*USA Economic Analysis Manual*).

Lines of Code (LOC): A measure of the size of computer software; a significant cost driver for development and maintenance; see also SLOC.

> Note: LOC as a size metric has been criticized as being ambiguous and even meaningless. While it is widely used, it may not be wise to rely on it as a sole measure of size. If a decision is made to use LOC, care should be taken to ensure it is consistently applied across projects and that differences in languages are taken into account.

Maintenance/Support: The process of modifying the existing operational software while leaving its primary functions intact. Software maintenance can be classified into two main categories: (1)

Update, which results in a changed functional specification for the software product, and (2) *Repair*, which leaves the functional specifications intact. In turn, *Repair* can be classified into three main subcategories: (2a) *Corrective maintenance* (of processing, performance, or implementation failures), (2b) *Adaptive maintenance* (to changes in the processing or data environment), and (2c) *Perfective maintenance* (for enhancing performance or maintainability) (Boehm 81).

Management Decision Package (MDEP): A structured life cycle process that represents the most current funding position developed through the Planning, Programming, Budgeting, and Execution System (PPBES). A separate MDEP will normally be created for each AIS. Each MDEP covers a nine-year period (*USA Economic Analysis Manual*).

Management Decision Process: A process for assimilating the results of the Technical and Economic Assessments (typically by technical questionnaire scores or economic indicators) to select the recommended strategy for each program and for prioritizing the list of programs.

Migration System: "An existing automated information system (AIS), or a planned and approved AIS, that has been officially designed as the single AIS to support standard processes for a function." "A migration system is designated (or selected) by the OSD Principal Staff Assistant(s) and their Defense Component counterparts whose function(s) the system supports, with the coordination of the DoD Senior Information Managements Official." From Deputy Secretary of Defense (memorandum dated October 13, 1993 subj: "Accelerated Implementation of Migration Systems, Data Standards, and Process Improvement," page 4).

Net Present Value: Traditionally, the difference between the present value of the dollar quantifiable benefits and the present value of the cost. (*USA Economic Analysis Manual*). It indicates the "net benefit or cost" of an alternative compared to another alternative.

New Development: The engineering process of developing new software by starting over at the requirements analysis phase without reusing existing requirements specifications, design, or code. An extreme case of redevelopment rarely occurring.

Non-Developmental Item (NDI): Items not requiring development *(DoDI 5000.2)*.

Non-Quantifiable Benefit: A benefit that is not adequately reflected using mathematical or numeric representation, such as better quality of services, and which is typically better expressed in narrative form (*USA Economic Analysis Manual*).

Parametric Cost Estimating Model: An aggregate of cost estimating relationships (CERs) which calculates predicted development and maintenance costs based on technical and schedule cost drivers.

Preferred Strategy: Ideally, a candidate strategy which has the highest Net Present Value (NPV) and/or the highest Benefit Investment Ratio (BIR).

Present Value Analysis: An analysis of competing alternatives using their present value.

Present Value: Dollars that have had their annual cash flow over time converted to equivalent amounts at a common point in time in order to account for the time value of money. The discount rate is prescribed by OMB. The computation begins with constant dollars which are multiplied by the discount rate (*USA Economics Analysis Manual*).

Program: Computer software (source code, assembly code, or code implemented as firmware).

Program Objective Memorandum (POM): See Management Decision Package, but for a five-year period.

Quantifiable Benefit: A benefit that can be assigned a numeric value such as dollars, physical count of tangible items, or percentage change (*USA Economic Analysis Manual*).

Ranking (or Rank Ordering): The relative ordering of candidate strategies based on an economic or technical metric (e.g., rank order #1 would be assigned to the most economically beneficial strategy).

Rate of Return (ROR): The interest rate at which the present value of the savings (benefit) equals the present value of the investment cost through the remaining life cycle of the alternative being evaluated (*USA Economic Analysis Manual*).

Recommended Strategy: The candidate strategy which is selected in the Management Decision Process.

Redevelopment: The engineering process of developing a new system by starting over at the preliminary design phase and reusing the existing requirements specifications.

Redocumentation: The process of analyzing the system to produce support documentation in various forms including users manuals and reformatting the system's source code listings (Santa Barbara I).

> Note: Sometimes, the output of reverse engineering is thought to be the same as redocumentation. After all, when reverse engineering captures the design information from the existing source code, the resulting display can take the form of data flow diagrams, control flow charts, etc. The difference between redocumentation and reverse engineering is that the redocumentation usually generates system documentation according to a standard. For example, there are redocumentation tools that create documentation for *MIL-STD-2167A*, the standard for documenting DoD software development projects.

Reengineering: The examination and alteration of an existing subject system to reconstitute it in a new form. The process encompasses a combination of subprocesses (strategies) such as reverse engineering, translation, restructuring, data reengineering, redocumentation, forward engineering, and retargeting, (Santa Barbara I).

Remaining Life (years): The required lifetime, starting when the reengineering effort begins. The sum of the reengineering time plus the operational lifetime of the reengineered software.

Restructuring: The engineering process of transforming the existing system from one representation form to another at the same relative abstraction level, while preserving the subject system's external functional behavior (Santa Barbara I).

> Note: In contrast to reverse engineering (which takes in source code and extracts design information which is a higher level of abstraction than the code), restructuring code leaves the system at the same abstraction level (within the life cycle) of "code," but radically rearranges the source code to fit a new paradigm such as structured analysis or object-oriented analysis.
>
> Restructuring is commonly used to mean taking unstructured code and making it structured. Architecture level changes need their own term and include changes to interfaces of modules, reduction of global data, remodularization of functionality, etc.

Retargeting: The engineering process of transforming, rehosting, or porting the existing system in a new configuration (Santa Barbara I).

> Note: The new configuration could be a new hardware platform, a new operating system, or a Computer Automated Support Environment (CASE) platform. In some cases utilization of CASE tools will not require a change of platform and the term retargeting would be misleading in cases where only a software change in the support environment is performed.

Retirement: Discontinuing use of a system, without revision (Santa Barbara I).

Reverse Engineering: The engineering process of understanding, analyzing, and abstracting the system to a new form at a higher abstraction level (Santa Barbara I).

> Note: This higher abstraction level is understood within the context of the software system's life cycle. For example, the classic waterfall development life cycle calls for requirements/specifications followed by design, code, test, implement, and maintain. Thus, if we start with the code, reverse engineering will extract design information which is at a higher abstraction level in the life cycle.

Source Lines of Code (SLOC*):* A metric for software size. Generally defined as executable source code statements. Specific definitions vary among parametric software estimating models, and depend on the implementing language (See Lines of Code).
Source Code Translation: Transformation of source code from one language to another or from one version of a language to another version of the same language (e.g., going from COBOL-74 to COBOL-85 or from CMS-2 to Ada) (Santa Barbara I).

Status Quo: Continued maintenance (support) of the existing software. In the Economic Assessment, this is always Strategy #1 (Santa Barbara I).

"Sunk" Costs: Past expenditures or irrevocably committed costs which are unavoidable and, therefore, should not be considered in the decision process (*USA Economic Analysis Manual*).

Systems Engineering: The top level process of engineering a system (hardware and software) to meet overall requirements.

Technically Acceptable Strategy: A strategy identified by the Technical Assessment Process as a candidate strategy for addressing the reengineering requirement. It must be a strategy that meets all related technical requirements.

Technical Assessment: A process for screening and evaluating attributes of an existing computer program to identify candidate strategies. These candidate strategies then will be economically assessed.

Technical Indicators: Metrics, derived during the technical assessment, that identify which candidate strategies have comparatively the greatest technical need to reengineer.

Then-Year Dollars: See Current Dollars.

Total Benefit (TB): The difference between the total O&S cost for Strategy 1 and the total O&S cost for Strategy N (including the O&S cost for Strategy 1 during reengineering).

Total Cost (TC): The sum of investment (CES 1.0) and Operations & Support (CES 2.0) cost for the remaining life of the software, excluding sunk costs. For reengineered software, Total Cost includes the O&S cost to maintain the existing software during reengineering. The analysis must include the continued use of the existing system until replaced by the alternative as part of the cost of the alternative (*USA Economics Analysis Manual*, paragraph 2-10).

Uniform Annual Cost: A constant dollar amount which, if paid annually throughout the economic life of a proposed alternative, would yield a total discounted cost equal to the actual present value of the alternative. It is determined by dividing the total discounted alternative cost by the sum of the discount factors for the years which an alternative yields benefits. It is used to compare alternatives with different economic lives (*USA Economic Analysis Manual*).

4. Reengineering Technical Assessment

4.1 Introduction

The Reengineering Technical Assessment (RTA) is the first part of the Software Reengineering Assessment Process. The primary purpose of this section is to match existing software components (otherwise known as "legacy" software or "candidate" software systems, programs, subroutines, etc.) with appropriate software reengineering or maintenance strategies. RTA is primarily focused on the technical aspects of legacy software systems. Therefore, it is recommended that this section be completed by the organization's technical personnel. RTA results are then passed on to Section 6 of this handbook to aid in management's reengineering decisions.

This handbook currently recognizes eight basic software maintenance strategies. Six of these strategies are considered reengineering strategies and two are classic maintenance strategies.

The six reengineering strategies are as follows:

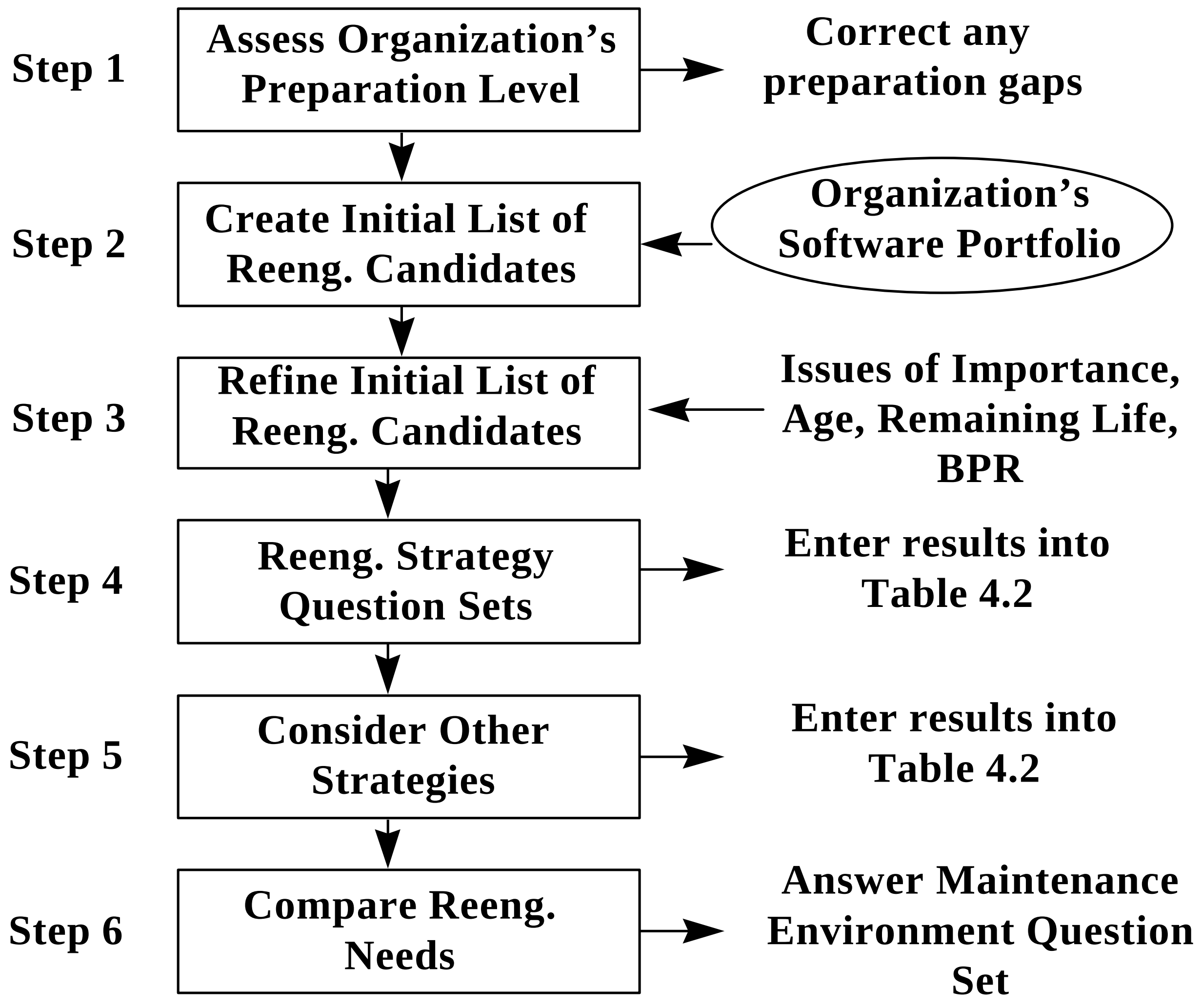
- Redocument
- Reverse Engineer
- Translate Source Code
- Data Reengineer
- Restructure
- Retarget

The two classic maintenance strategies are as follows:

- Redevelopment
- Status Quo

The RTA will provide a quick and easy method for determining which of the above eight software maintenance strategies is best suited to a given candidate software component.

Figure 4-1
Reengineering Technical Assessment
Flow Chart


Step 1
Assess Organization's Preparation Level
Correct any preparation gaps
Step 2
Create Initial List of Reeng. Candidates
Organization's Software Portfolio
Step 3
Refine Initial List of Reeng. Candidates
Issues of Importance, Age, Remaining Life, BPR
Step 4
Reeng. Strategy Question Sets
Enter results into Table 4.2
Step 5
Consider Other Strategies
Enter results into Table 4.2
Step 6
Compare Reeng. Needs
Answer Maintenance Environment Question Set

4.2 Step One: Assess the Organization's Level of Preparation

By studying successful and unsuccessful software reengineering projects, this handbook has determined the minimum criteria an organization should meet to help ensure a successful reengineering project.

The question set entitled "Reengineering Preparation" beginning on page 4-14, has been created to help assess an organization's level of preparedness to reengineer. By adding up the answers of each question and dividing by the number of questions answered, determine the average response per question (use the RTA Summary Worksheet Table 4.2).

If the average is below 2.00, this handbook strongly recommends that an organization correct its level of preparation using this question set as a rough guide for what needs to be addressed. If the average is between 2.00 and 3.00, circle "yes" next to "Preparation" on RTA Summary Worksheet 4.2 for the candidate software. Otherwise circle "no" next to "Preparation" on the RTA for the candidate software. ***Do this before using any additional part of this handbook!*** Without a high level of preparation, the reengineered software will, at best, require another round of reengineering in a few years and, at worst, the reengineering project will fail.

4.3 Step Two: Identify Software Reengineering Candidates

From the portfolio of all the software systems within the organization[1], identify and list those systems, programs, or components which are perceived as being a problem, or having significant improvement potential. Look at the organization's base of existing software with the following issues in mind:

- Age - is the software very old?
- Complexity - is the software too complex (using complexity measurement tools)?
- Language - is the language out of date, not portable, or cumbersome to maintain?
- Reliability - does the software fail periodically?
- Level of maintenance required (in dollars or man-hours) - is the software difficult or expensive to maintain?
- State of documentation - is the documentation poor and/or inadequate?
- Impact if software should fail - will the customer be severely impacted by the software's failure?
- Degree of coupling between hardware and software - is the software's coupling with the hardware blocking future enhancements?

[1]This assumes there exists a current portfolio of all software systems in use. If not, create one prior to this step.

- Personnel knowledge, experience, and turnover - is there currently (or in the near future) a lack of experienced maintenance programmers for this software?
- Technological constraints - is the software or hardware blocking future enhancements?
- Change of platform...past or future - has a change of platform occurred for the software or will a change soon occur?

4.4 Step Three: Reduce the List of Software from Step Two

The list of candidate reengineering software from step two may still be long. *Remove a software candidate from the above list if any of the following criteria apply:*

- The remaining life of the candidate software is projected to be less than 3 years.

  Reengineering usually requires significant start-up resources (personnel, training, software, and hardware). These reengineering costs can be amortized over the remaining life of the software since software reengineering will reduce annual software maintenance costs. Typically, at least three years are required before an organization will see a return on its reengineering investment. Therefore, if remaining life is greater than three years, this handbook suggests circling "yes" next to "Remaining Life" on RTA Summary Worksheet for the candidate software. If not, circle "no" next to "Remaining Life" on RTA Summary Worksheet for the candidate software.

- The candidate software isn't important enough to reengineer.

  The candidate software should be evaluated against the question set "Software Importance". The average of answers for this question set should fall within the range of 2.00 to 3.00 to meet this importance criteria. If so, circle "yes" next to "Importance" on RTA Summary Worksheet 4.2 for the candidate software. If the average is less than 2.00, drop the software from the list of candidate reengineering software and circle "no" next to "Importance" on RTA Summary Worksheet 4.2 for the candidate software.

- The candidate software was first developed less than 5 years ago.

  This assumes modern programming techniques (such as structured programming or object-oriented design) are now being used and have been used over the past five years. Newer software has the additional advantage of access to the original development staff who understand the software for easier maintenance. Reengineering shows the best return when used on candidate software that is difficult to maintain. Therefore, if the age of the software is greater than five years, this handbook suggests circling "yes" next to "Age" on RTA Summary Worksheet 4.2 for the candidate software. If not, circle "no" next to "Age" on RTA Summary Worksheet 4.2 for the candidate software.

- The candidate software directly supports a business process currently being reengineered.

  If the organization is undergoing business process reengineering (BPR) and the candidate software directly supports a process being reengineered, then don't reengineer that software until it is determined what software changes are needed. BPR will fundamentally change the way an organization functions. Software reengineering supports BPR. Thus, any software that supports a business process undergoing BPR will also change fundamentally. The final process must be fully defined before changing the software that supports that process.

Candidate software, rejected for reengineering due to one or more of these criteria, may be retained on the list of reengineering candidates if there are extenuating circumstances (management demands, customer pressure, funding source restrictions, etc.). Additional criteria, unique to each organization, may also be applied to the list from step two in order to reduce the number of candidates for reengineering.

4.5 Step Four: Complete the Reengineering Technical Assessment RTA Questions

For each remaining candidate software, answer the question sets corresponding to the following reengineering strategies (create copies of the questions sets for each candidate software component to be evaluated for reengineering):

- Redocument - page 4-19
- Translate Source Code - page 4-20
- Data Reengineer - page 4-22
- Retarget - page 4-24
- Restructure - page 4-26

Consult with personnel familiar with the candidate software. Group consensus helps to minimize potential error or bias. After answering as many of the questions as possible, compute the total for each question set and divide by the number of questions answered (use the accompanying RTA Summary Worksheet). This average response for each question set will determine whether that reengineering strategy is a good match for the candidate software being considered. A few skipped questions, which cannot be answered, will not significantly affect the reengineering assessment score since other questions are designed to elicit similar information from a different perspective.

- If the average falls within the range of 2.40 to 3.00, then that reengineering strategy is a very good match for the candidate software. Circle "yes" under "Indication of Strategy" for that strategy.

- If the average falls within the range of 1.70 to 2.39, then some benefit is indicated by applying that reengineering strategy but the match is not solid. There may be other strategies better suited to this candidate software. Circle "maybe" under "Indication of Strategy" for that strategy.

- If the average falls within the range of 1.00 to 1.69, then the reengineering strategy is a poor match for the candidate software. Circle "no" under "Indication of Strategy" for that strategy.

4.6 Step Five: Consider Other Strategies

Note that the RTA Summary Worksheet includes three additional strategies. This section will explain when these strategies should be considered.

4.6.1 Reverse Engineer

Reverse engineering is the process of extracting design information from source code, storing the extracted information in a repository, and graphically presenting this information (such as a control flow diagram, data flow diagram, etc.). Reverse engineering is a larger, more complex, and usually more expensive strategy than any of the other five reengineering strategies alone. Many organizations that decide to reverse engineer use the resulting design information repository as a key element to their long term maintenance process. Such organizations maintain their software by modifying the repository information instead of the source code. The executive Summary Worksheet source code can then be generated from the altered design information.

Since reverse engineering is a larger, more complex, and usually more expensive strategy, it is normally used only when one of the following conditions are met:

- Multiple reengineering strategies are indicated.

  Reverse engineering can be used as a common, interim step when two or more reengineering strategies are indicated from RTA Summary Worksheet. For example, if restructuring and source code translation are indicated, reverse engineering could facilitate the other two strategies in the following way:

  First, capture the design information using reverse engineering. Understand how the existing software functions. Clean up the candidate source code at the design level, eliminating any "dead" code (non-executable). Second, generate the new source code language from the improved design information. Restructure the generated source code using restructuring tools created by vendors for the new source code language. Third, capture the restructured design information back to a design repository.

This process allows the maintenance programming staff to always understand how the software works even after it's translated to a new language and the control flow is radically altered by restructuring.

- The control flow of the candidate software is too complex to understand

  The documentation is missing or wrong and years of haphazard maintenance has created software that cannot be modified without creating unwanted side-effects. Studies show that software understanding is 70% of most maintenance activities. As described in the previous bullet, reverse engineering allows the maintenance staff to understand the control flow and eliminate unused code.

- Reuse is a key objective of the organization

  Source code repositories have always been key to an organization's reuse strategy. However, the reusable code fragments stored in such repositories are usually not fully trusted by the development staff. They do not know exactly how each reusable code fragment functions, and how it will interface with the rest of their new software. Although there's documentation, it's questionable whether it's accurate, complete, and current.

  On the other hand, a design repository's reusable *design* fragments will inherently show the developers what's going on inside each fragment. Such fragments can be combined and altered to create a larger software design. The resulting design can then be translated into executable source code using forward engineering tools.

In conclusion, the reverse engineering strategy is indicated if any of the preceding conditions are met. If so, circle "yes" next to "Reverse Engineer" on RTA Summary Worksheet for the candidate software. Otherwise, circle "no" next to "Reverse Engineer" on RTA Summary Worksheet for the candidate software.

4.6.2 Redevelopment

Redevelopment is the traditional, non-reengineering solution for really bad source code. The Redevelopment option is used when the candidate software (including its design) is not worth salvaging and should be developed from scratch. If three or more reengineering strategies are indicated using the associated question sets, and the projected remaining life of the candidate software exceeds five years, then seriously consider redevelopment for the candidate software. Three or more reengineering strategies may prove as expensive as redevelopment and may indicate that the software is not worth salvaging by reengineering techniques.

There are certain advantages to redevelopment. "Getting it right this time" using modern programming practices certainly leads the list. Of course, this assumes the organization's current development and maintenance practices are now well-defined and enforced. Maintenance costs will generally be significantly lower once the redevelopment is completed. Lower even than if this software had been reengineered instead of redeveloped.

The disadvantages reflect a significantly higher cost to redevelop as compared to reengineering. That is why five years is generally considered the minimum number of years to recoup redevelopment investment due to much lower maintenance costs (similar to the reason why "newer" programs of less than five years since development are generally not considered as candidates for reengineering).

Thus, if both conditions are met (three or more reengineering strategies are indicated *and* the remaining life is more than five years), circle "yes" next to "Redevelop" on RTA Summary Worksheet for the candidate software. Otherwise, circle "no" next to "Redevelop" on RTA Summary Worksheet for the candidate software.

4.6.3 Status Quo

If the question sets determine that no reengineering strategy is needed, then leaving the software alone may be the best policy. If this is the case for the candidate software, circle "yes" next to "Status Quo" on the RTA Summary Worksheet. If any reengineering strategy is indicated, circle "no" next to "Status Quo".

4.7 Step Six : Comparing Reengineering Needs Across Different Software Components

Now that the reengineering needs of a given software component have been assessed, different software components (and their respective reengineering strategies) must be compared. For example, software component 'A' may indicate a need to restructure and retarget while software component 'B' may indicate a need to translate source code and data reengineer. How does component 'A' and its reengineering needs compare to component 'B' and its reengineering needs?

4.7.1 Management Decision Aid.

Enter the reengineering strategy question set results from RTA Summary Worksheet into the management decision worksheet found in Section 6 of this handbook. This technical assessment information will be combined with the cost data for each reengineering strategy (Section 5) and then both will be considered by management (Section 6).

4.7.2 Maintenance Environment

As an additional factor for comparing reengineering needs between software components, answer the Maintenance Environment Question Set on page 4-28. This set of questions assesses the current maintenance environment of the candidate software. Since reengineering generally results in software that is more easily maintained, this question set considers other factors that may raise (or lower) the urgency to reengineer a particular software candidate.

If the average of answers for this question set falls within the range of 2.00 to 3.00, circle "yes" next to "Maintenance Environment" on RTA Summary Worksheet for the candidate software. If the average is less than 2.00, circle "no" next to "Maintenance Environment" on RTA Summary Worksheet for the candidate software.

4.8 An Example Using the SRAH Reengineering Technical Assessment

The following is an example of applying the SRAH Reengineering Technical Assessment Process.

Step one: After answering only 24 of the Reengineering Preparation questions, you add up the 24 answers and discover a total of 59 points. Divided by 24, the average answer is 2.46. Since 2.46 is above the 2.00 threshold, you decide that your organization has done sufficient planning (i.e. is prepared) for a reengineering project. Thus, you circle "yes" next to Preparation in RTA Summary Worksheet. However, you notice a few questions in the Preparation Question Set in which you had to answer "1". Mark these for immediate attention and plan to correct them. Enter the numbers 24, 59, and 2.46 within their respective columns for Preparation in worksheet 4.2. These numbers will be the same for all candidate software components so you don't have to re-answer the Preparation Question Set.

Step two: Based on criteria listed in section 4.3, compile a list of 33 software components that warrant attention. Some of these software components are maintenance nightmares, others are very old systems that are inhibiting your organization from taking advantage of newer hardware platforms or CASE tools.

Step three: Of the 33 "problem" software candidates, you discover 3 were developed only 2 years ago. (Make a mental note to discover why these relatively new software components were so poorly developed). Another 6 are due to be discontinued within a few years due to customer disinterest or future funding constraints. While 5 more directly support the accounting system which will soon be overhauled due to an upcoming reorganization of that business sector. This leaves 19 candidate software components (33 minus 14 equals 19).

By applying the Importance Question Set, against each of the remaining 19 candidate software components, you discover that only 7 are important enough to spend critical organizational resources (personnel and funds) to warrant reengineering. Of these 7, you begin with candidate software component "A". For software component "A", the sum of your 5 answers for the Software Importance Question Set equals 13 for an average of 2.6. So you circle "yes" next to Importance in RTA Summary Worksheet and write a brief description of software component "A" in the upper left corner of this worksheet. Enter the numbers of 5, 13, and 2.6 within their appropriate columns for Importance on RTA Summary Worksheet.

Having passed the Age and Remaining Life criteria, circle "yes" on RTA Summary Worksheet for these two items for software component "A".

Step four: For software component "A", answer the question sets corresponding to the five reengineering strategies of redocument, data reengineer, restructure, translate source code, and retarget.

Results:

redocument:
- 5 questions answered, sum of 14, average of 2.8
- circle "yes"

translate source code:
- 6 questions answered, sum of 15, average of 2.5
- circle "yes"

data reengineer:
- 9 questions answered, sum of 12, average of 1.33
- circle "no"

retarget:
- 5 questions answered, sum of 9, average of 1.8
- circle "maybe"

restructure:

6 questions answered, sum of 9, average of 1.5
circle "no"

Step five: You now need to consider whether reverse engineering, redevelopment, or status quo are valid strategies as well.

Reverse engineer:

Two reengineering strategies were indicated (redocument and translate source code) and another was possibly indicated (retarget). Since two "yes" indicate that reverse engineering should be considered, circle "yes" next to this strategy on the worksheet. Your strategy may be to reverse engineer the candidate software, thus capturing its control flow and other design information. This design information can now be used to document the candidate software. When translating source code, you need to take advantage of the power of the new language. To do this, you could manipulate the design information within the design repository then use a source code generator to create executable source code in the new language.

Redevelopment:

Two "yes" and one "maybe" indicate a borderline possibility of redevelopment. Decide whether to consider redevelopment when doing cost estimates in section 5 of this handbook.

Status Quo:

At least two reengineering strategies are indicated so circle "no".

You answered 7 of the Maintenance Environment questions for a total 12 points or an average of 1.71. Circle "no" next to "Maintenance Environment" on RTA Summary Worksheet since the average is less than 2.00.

Repeat this process, using copies of the worksheet, for the other six candidate software components. Transfer the results to the appropriate worksheets in Sections 5 and 6.

4.9 The Importance of a Pilot Project

As with any new technology, reengineering should be inserted cautiously within an organization. After choosing the prime software candidates for reengineering, choose one that is small enough and isolated enough (little or no impact on other software components and data files) to make a good pilot project. Such a pilot project enables an organization to test the newly acquired tools and newly defined techniques of reengineering. Lessons learned from pilot projects can then be applied to larger, more complex software reengineering projects.

4.10 Impact Analysis

Several reengineering strategies will substantially change existing source code. Since many software components do not stand alone, management must be advised of the impact of reengineering on dependent software components and data files.

For example, source code translation from COBOL to Ada will impact how the translated software interfaces with its data files. If those data files are IMS and the existing software uses embedded IMS macro calls, then the organization must consider how the newly translated software will now access those same data files. Such considerations may require data reengineering of the existing data files to a format accessible by the new software. IMS-like macro calls for the Ada compiler may exist. If the organization ends up data reengineering the existing data files, what impact will this have on other software systems that use these same data files?

Therefore, this handbook recommends listing and defining all data files and dependent software systems that interface with the reengineering candidate software. Carefully explain how each reengineering candidate accesses its data files and passes information to other software systems. In this way, management will understand the full scope of the proposed reengineering projects.

4.11 Additional Reengineering Considerations

Two reengineering strategies may have some difficulty if implemented alone. These strategies are restructuring and source code translation.

Restructuring will radically alter the internal control flow of the existing software. The maintenance staff, currently familiar with the control flow of the existing software, will have initial difficulty understanding the newly restructured software. Thus, an experienced maintenance programmer and one unfamiliar with the software will be on the same footing. Some organizations reassign such restructured code to a new set of maintenance programmers. This handbook recommends that some form of reverse engineering accompany restructuring of source code. A graphical representation of the control flow, before and after the restructuring, will facilitate software understanding and future maintenance requests.

Source code translators typically perform line-for-line translations. This approach does not take advantage of the advanced semantic constructs inherent in the new language. For example, when translating source code from COBOL to Ada, line-for-line translation will not take advantage of Ada's object-oriented constructs. This handbook recommends restructuring take place *after* the source code translation. Restructuring tools are written with specific languages in mind. Thus, most restructuring tools will help adapt the translated code to the new language. Consider reverse engineering as an interim step for source code translation. After reverse engineering, one could manipulate the graphical design representation to fit the new language's constructs, then generate the new source code from the design repository.

4.12 Software Analysis Tools

All reengineering strategies should include tools for analysis of candidate software prior to reengineering. Analysis tools can:

- Identify "dead" (unused) code
- Measure the candidate software's level of complexity
- Identify poor coding practices (such as uninitialized variables, etc.)
- Measure the number of function points
- Check for non-structured constructs

Cleaning up the candidate software prior to reengineering will facilitate the reengineering project by eliminating problem code before subjecting the software to the reengineering process. It makes no sense to translate, restructure, redocument, reverse engineer, or retarget bad source code embedded within the candidate software.

4.13 Reporting the Results of this RTA

All the work accomplished by the technical staff during this RTA should be reported to management using section 6 of this handbook. RTA Summary Worksheet will be used by Sections 5 and 6 of this handbook. Additionally, recommendations based on sections 4.9 through 4.12 should also be passed along to management. Write a short report containing these technical assessment findings as an addendum to the final report prepared for management in Section 6. This addendum should include the results of RTA Summary Worksheet and Sections 4.9-4.12 with a brief explanation for each finding.

TABLE 4-1 REENGINEERING TECHNICAL ASSESSMENT QUESTIONS

REENGINEERING PREPARATION

Organizational Needs

1. Have the purposes/goals of reengineering been defined for this reengineering project?

   1 - No
   2 - Yes but they're unclear to me or don't know them
   3 - Yes

2. Does management support this reengineering effort?

   1 - No
   2 - Yes but with reservations or management perceives that reengineering will solve all their maintenance problems
   3 - Yes

3. How important is this reengineering project to the organization (managers, users, etc.)?

   1 - Not important
   2 - Average importance
   3 - Critical

4. Is there financial support for this reengineering effort?

   1 - Expect little or no financial support
   2 - Expect some funding but not sufficiently funded
   3 - Expect project will be fully funded

5. Is the reengineering effort consistent with overall corporate strategy?

   1 - Not aware of a corporate strategy
   2 - Mostly
   3 - Yes

Reengineering Team

6. Has a reengineering team been chosen? (Picked from the users, maintenance personnel and management)

   1 - No or not representative of all key groups
   2 - Yes but not dedicated full time
   3 - Yes

New Maintenance Process Defined

7. Has a new maintenance process environment been clearly documented?

1 - No
2 - Yes but only in broad, general terms
3 - Yes

8. Has an organizational CMM software assessment been done?

1 - No
2 - Yes but unknown
3 - Yes

9. The organization's CMM level is:

1 - One
2 - Two
3 - Three or above

10. Has a tactical plan been defined and implemented to improve the CMM level of the maintenance organization?

1 - No tactical plan to improve maintenance
2 - Yes, the tactical plan has been defined
3 - Yes, the tactical plan is being implemented

Metrics

11. Have development/maintenance software metrics been defined for the organization?

1 - No
2 - Yes but the metrics are not based on process improvement or related to corporate goals
3 - Yes

12. Are software metrics being used regularly?

1 - No
2 - Unsure how metric information is being applied
3 - Metrics play a key role in shaping maintenance decisions

13. How will metrics be used on the reengineering project?

1 - Will not be used
2 - Metrics still being defined or unsure of metrics usage
3 - Metrics will determine whether the reengineering project was successful

14. Has a metric been defined to measure software maintainability (such as manhours per SLOC changed)?

    1 - No
    2 - Not sure or metric is not entirely representative
    3 - Yes

Reengineering Process Defined

15. Has a reengineering process been chosen? (Incremental or complete systems)

    1 - No
    2 - Not sure which reengineering process is best yet
    3 - Yes

16. Is reuse part of the organization's reengineering goals?

    1 - Reuse has not even been considered
    2 - Reuse has been considered but is tangential to this reengineering project
    3 - Reuse is a key element of this reengineering project

17. How will the software be maintained while reengineering occurs?

    1 - Not been considered
    2 - Will add modifications to both reengineered software and old software simultaneously
    3 - Configuration management will log all modifications to old software and then will roll these into the completed reengineered software

Standard Testbed or Validation Suite

18. Is there a current software validation test suite and is it complete?

    1 - No standard validation suite
    2 - Yes, but don't know if it's current and complete
    3 - Yes

19. Is regression testing used during maintenance changes?

    1 - Rarely
    2 - Usually but not consistently
    3 - Yes, always

20. Is the testing strategy fully documented?

    1 - No
    2 - Yes, but unsure if testing documentation is complete or fully accurate
    3 - Yes

21. How will the reengineered software be validated?

    1 - Not considered this
    2 - Will use current validation test suite
    3 - Detailed test plan, including functional traceability

22. Does the new maintenance environment include testing?

    1 - No
    2 - Testing is included but is mostly ad hoc or blackbox
    3 - Testing is an integral part of the new maintenance environment

Tools Analysis

23. Are tools available (to automate the various reengineering strategies) for the target hardware platform and source code language?

    1 - No, or don't know
    2 - Yes, but would require extensive customization
    3 - Yes

Training

24. How much current training/understanding of reengineering is there within the maintenance organization?

    1 - Few, if any, of the reengineering team have been trained
    2 - Some of the reengineering team have been trained
    3 - Most of the key reengineering team have been trained in reengineering, tools, and the target maintenance environment

25. Who is to be trained and in what area?

    1 - Training has not been considered
    2 - Training is part of this reengineering project but not in detail
    3 - People have been designated and specific classes identified

26. Is there a funded plan to train the reengineering staff in any new tools or techniques which will be used?

    1 - No
    2 - Yes, but less than 10% of reengineering budget
    3 - Training represents 10% or more of the entire reengineering budget

SOFTWARE IMPORTANCE

1. How much does the candidate software contribute to the customer's mission?

   1 - Minimally
   2 - Somewhat
   3 - Significantly

2. If the candidate software failed what would be the effect?

   1 - Little or no damage
   2 - Significant damage
   3 - Permanent damage, major financial loss, or loss of life

3. Is there a contingency plan (current, recently tested, and ready at a moment's notice) which could be used if the candidate software fails?

   1 - Yes, or not needed
   2 - Yes, but with some difficulty and a significant loss of efficiency
   3 - No

4. If the candidate software failed, would it effect other mission-critical systems or data files?

   1 - Little or no effect on other mission-critical systems or data files
   2 - Significant disruption to other mission-critical systems or data files
   3 - Permanent damage, major financial loss, or loss of life due to disruption of other mission-critical systems or data files

5. Is the candidate software important to the organization's management?

   1 - No
   2 - Somewhat, or don't know
   3 - Yes

6. Is the candidate software important to the customers?

   1 - No
   2 - Somewhat, or don't know
   3 - Yes

7. Is the software being used?

   1 - No, the staff has developed small systems or manual processes to replace the software's function
   2 - To some extent, but there are valid complaints about the software's functional shortcomings
   3 - Yes, the software performs the business functions it was meant to fulfill

	8. Does the candidate software have a maintenance backlog? 1 - No 2 - Yes, but steady or decreasing 3 - Yes, and increasing 9. Characterize the annual change traffic* of the candidate software within the last 12 months? 1 - Small or insignificant 2 - Moderate 3 - Large or significant * Annual Change Traffic is defined as "the fraction of the software product's source instructions which undergo change during a year, either through addition, deletion, or modification."

REDOCUMENT

1. The candidate software's documentation is best characterized as:

   1 - Complete and current
   2 - Incomplete or not current
   3 - Non-existent or misleading

2. If the candidate software changes, is the documentation updated after each change?

   1 - Always
   2 - Sometimes
   3 - Rarely or never

3. Is the technical documentation useful and used during maintenance activities?

   1 - Yes, the maintenance staff can easily use the documentation to understand the candidate software
   2 - Using the documentation can be time-consuming
   3 - No, the documentation is non-existent or cumbersome and difficult to use

4. Does the documentation follow a standard format (such as MIL-STD-2167A or organization-specific)?

   1 - Yes
   2 - Yes, but standard is vague or documentation does not completely comply
   3 - No standard

5. Throughout the lifetime of the candidate software, has there always been an effective enforcement process to ensure adherence to the organization's documentation standards?

   1 - Yes
   2 - The enforcement process is not consistently applied or monitored
   3 - No enforcement process

TRANSLATE SOURCE CODE

1. What is the candidate software's principal language level?

    1 - Advanced, high level languages such as Ada, C++, Focus, etc.
    2 - Higher order languages such as FORTRAN, COBOL, C, Pascal, etc.
    3 - Assembly languages

2. How many programming languages does the candidate software use?

Note: Embedding SQL or similar database language within a High Order Language does not count as two separate languages. Rationale: Using embedded database calls does not increase the need to reengineer the program.

    1 - One
    2 - Two
    3 - Three or more

3. Is there a need to translate/consolidate the source code language(s)?

    1 - No need
    2 - Not yet but anticipate a mandate soon
    3 - Strong need due to external mandate (such as Ada) or to satisfy reengineering goals

4. Is there personnel trained in the new target language?

    1 - No
    2 - Not currently but there is a defined, formal plan to hire/train personnel
    3 - Yes

5. Is the language currently used by the candidate software based on a proprietary compiler?

    1 - The compiler is based on a DoD-approved high order language (HOL) with strict acceptance standards
    2 - The compiler has had some minor deviations/enhancements from an approved HOL
    3 - The compiler has had major modifications deviating from an approved HOL, making the compiler highly proprietary

6. Are sufficient support tools available for the language currently used by the candidate software?

    1 - Yes, for the current language there exist configuration management tools, development tools, maintenance tools, analysis tools, and a variety of compilers
    2 - There exist a limited number of support tools for the current language. Some of these may have been developed in-house
    3 - No, there are very few support tools for the existing language

	7. How portable is the language currently used by the candidate software? 1 - Language is very portable 2 - Portable, but significant embedded routines that are not portable 3 - Not very portable to other platforms 8. Is the language currently used by the candidate software mostly written in an obsolete dialect of that language? 1 - The language is the most current dialect 2 - Mixture of dialects of same language is used by the candidate software 3 - The language uses an obsolete dialect

DATA REENGINEER

1. Characterize the data files:

   1 - Data is managed by a modern data file organization (such as relational or object-oriented databases)
   2 - Some of the data is managed by a modern data file system, but other files use old formats (such as flat files, VSAM, etc.)
   3 - Data is mostly managed by old file formats (such as flat files, VSAM, etc.)

2. Are the relationships between data elements well-documented?

   1 - Yes
   2 - Not consistently
   3 - Data relationships not clearly defined or understood

3. How old are the current data files?

   1 - Less than 3 years
   2 - Three to five years
   3 - More than 5 years

4. How many different data files?

   1 - One to three
   2 - Three to five
   3 - More than five

5. Is the candidate database hindering migration to more advanced technology?

   1 - No
   2 - Some data files create a problem
   3 - Yes

6. Are the data files used by other systems?

   1 - Yes
   2 - Yes, but only one or two files are shared with other systems
   3 - No, the files to be data reengineered are exclusively used by a single system

7. Does the software maintenance organization change modules or just develop new ones when functionality changes? (bearing on data name rationalization)

   1 - Maintenance usually changes existing modules
   2 - Maintenance sometimes changes existing modules and sometimes creates new modules
   3 - Maintenance usually creates new modules

	8. Does the organization use a standard data dictionary? 1 - Yes 2 - Yes, but not strictly enforced and therefore not current 3 - No standard data dictionary 9. How many different programs within the candidate software? 1 - One to three 2 - Three to seven 3 - Over seven 10. If any recently reengineered/redeveloped software uses the candidate data file, what is the impact to the candidate data file? 1 - Insignificant or no impact to candidate data file 2 - Minimal impact to candidate data file 3 - The candidate data file is critical to the reengineered software and will be significantly affected by the software reengineering project

RETARGET

1. How portable is the language currently used by the candidate software?

   1 - Language is very portable
   2 - Portable but significant embedded routines that are not portable
   3 - Not very portable to other platforms

2. Is the candidate software stand-alone or is it a part of a larger system?

   1 - Candidate software is closely connected to a larger system
   2 - Candidate software is connected to a larger system but candidate software could be isolated with a well-defined interface
   3 - No, candidate is a stand-alone

3. What is the age of the current hardware platform?

   1 - Less than 3 years old
   2 - Three to six years old
   3 - Over six years old

4. Does the organization need to maintain duplicates of the same candidate software due to different hardware platforms required for the software to execute upon?

   1 - No
   2 - Not sure, or differences are not significant and easily maintainable
   3 - Yes

5. Does the new maintenance environment call for CASE tools?

   1 - No
   2 - Not currently, but will probably move to CASE tools soon
   3 - Yes

6. Does the current hardware platform cause severe execution problems?

   1 - No
   2 - Yes, but not frequently
   3 - Yes

7. Is vendor support for the current hardware guaranteed for the remaining software life?

   1 - Yes
   2 - No, but support loss is unlikely
   3 - No, and this is an immediate concern

	8. Is the current hardware platform limiting expansion or future development activities? 1 - No 2 - To some extent 3 - Yes, further software expansion, future development, or new technical advances are limited due to the existing hardware platform

RESTRUCTURE

1. What is the average number of Source Lines of Code (SLOC) per Computer Software Unit (CSU*) in the candidate software?

* A CSU is defined as the smallest cohesive software unit that is separately testable. A CSU usually corresponds to a called function or sub-routine within a program.

1 - Less than 50
2 - 50 and 200
3 - More than 200

2. What is the average number of Function Points per CSU of the candidate software?

1 - Less than 500
2 - 500 - 2500
3 - More than 2500

3. What is the average Cyclomatic complexity per module?

1 - Ten or less
2 - Between 10 and 20
3 - More than 20

4. What is the average Essential complexity per module?

1 - Five or less
2 - Between 5 and 10
3 - More than 10

5. The candidate software was created using a development process that was:

1 - Rigidly followed
2 - Sometimes followed
3 - Not followed or non-existent

6. The change control procedure for the candidate software has been:

1 - Strictly enforced
2 - Loosely enforced
3 - Ad hoc or does not exist

7. Characterize the annual change traffic* of the candidate software within the last 12 months?

   1 - Small or insignificant
   2 - Moderate
   3 - Large or significant

* Annual Change Traffic is defined as "the fraction of the software product's source instructions which undergo change during a year, either through addition, deletion, or modification."

8. Does software maintenance usually involve changing modules or developing new modules?

   1 - Maintenance usually is implemented by creating new modules
   2 - Maintenance uses approximate equal new modules vs. changing existing modules
   3 - Maintenance usually consists of changing existing modules

9. Is technology available for the analysis of the candidate software (complexity, function point analysis, etc.)?

   1 - No
   2 - Don't know, or not entirely accessible
   3 - Yes

10. Over the past two years, does the candidate software require more man-hours to modify per SLOC?

   1 - Maintenance man-hours per SLOC is low and constant
   2 - Maintenance man-hours per SLOC is increasing slowly
   3 - Maintenance man-hours per SLOC is increasing significantly

MAINTENANCE ENVIRONMENT

1. Will the size of the candidate software's support staff remain stable?

   1 - Yes
   2 - Minor staff changes are anticipated
   3 - Major staff changes/cutbacks are due

2. For the candidate software, are there sufficient maintenance programmers?

   1 - Yes, allowing timely maintenance and happy users
   2 - Current number of maintenance personnel is usually sufficient with occasional maintenance backlogs
   3 - No, thus causing a severe maintenance backlog and unhappy users

3. Is there sufficient maintenance personnel with in-depth experience on the candidate software?

   1 - Yes
   2 - Yes, but may be losing some in the near future
   3 - No

4. Characterize the maintenance personnel turnover (per year):

   1 - Very stable group of maintenance personnel
   2 - A few people leave each year
   3 - Heavy or frequent turnover

5. Are the original developers available for consultation?

   1 - Yes
   2 - Yes, but the candidate software is old or the developers are not easily accessible
   3 - No

6. If maintenance personnel for the candidate software leave, will they be replaced?

   1 - Yes
   2 - Some natural attrition is a goal of the organization
   3 - No

7. Is the candidate software more expensive (man-hours, productivity loss, etc.) to maintain when compared to the rest of the software maintained by the organization?

   1 - Less expensive than average
   2 - About average
   3 - More expensive than average

	8. How do the users view the candidate software's quality? 1 - Improving or satisfactory 2 - Remaining the same and barely tolerable 3 - Declining 9. Characterize the Mean Time Between Fixes (MTBF) for the candidate software (i.e. how often is a software error discovered): 1 - Few if any errors are detected 2 - Moderate number of errors 3 - Frequent errors with more created after each modification 10. After an error is detected, characterize the Mean Time To Repair (MTTR) the candidate software: 1 - Errors are fixed quickly 2 - Errors are added to the backlog and fixed within a reasonable amount of time 3 - The backlog requires a large amount of time before an error is addressed

Reengineering Technical Assessment
Table 4-2 RTA Summary Worksheet

Candidate Software Name & Description: ________ ________ ________	Number of Questions Answered	Summation of Answers	Average: (Summation Divided by Number of Questions)	Indication of Strategy
Remaining Life (>3 yrs?)				yes no
Age (> 5 yrs?)				yes no
Preparation[1]				yes no
Importance[1]				yes no
Maintenance Environment[1]				yes no
Redocument[2]				yes maybe no
Restructure[2]				yes maybe no
Translate Source Code[2]				yes maybe no
Data Reengineer[2]				yes maybe no
Retarget[2]				yes maybe no
Reverse Engineer				yes no
Redevelop				yes no
Status Quo				yes no

[1] The “yes” range is from 2.00 to 3.00. The “no” range is below 2.00.

[2] The “yes” range is from 2.40 to 3.00. The “maybe” range is from 1.70 to 2.39. The “no” range is below 1.70.

5. Reengineering Economic Assessment Process

5.1 Introduction

The Reengineering Economic Assessment (REA) process is the second part of the Software Reengineering Assessment (SRA) process. The REA process is performed on candidate software identified in the Reengineering Technical Assessment (RTA) process (Section 4). An REA is done on all the technically appropriate strategies being considered for a specific candidate. If more than one candidate is being evaluated, the REA is repeated for each candidate. Finally, in the Reengineering Management Decision (RMD) Process (Section 6) the results of all the RTAs and REAs are re-evaluated in terms of programmatic, organizational, and other non-economic issues to establish overall organizational priorities.

The REA is intended to be performed by one or more software analysts with an understanding of software economics. These personnel need to have a thorough understanding of the existing software, reengineering, cost estimation/modeling, and economic indicators, or have access to someone who has this knowledge. The purpose of the REA is to develop credible and consistent cost estimates for each strategy.

Although highly accurate cost estimates are always desirable (and the REA makes every attempt to display all costs accurately), the REA's focus is on the consistency of the estimates for purposes of decision making. The accuracy required is that which produces the same results as an assessment performed by one or more independent analysts. The preferred economic indicators used, Net Present Value (NPV) and Benefit Investment Ratio (BIR), are intended for comparing strategies on equal terms rather than producing a budgetary input. After selected strategies are identified and organizational priorities are established, detailed budget quality estimates can be developed in current year/then year (inflated) dollars for each proposed effort.

Finally, the REA is not intended to be a comprehensive tutorial on software estimating or economics, nor does it produce cost estimates that can be used to support a budget or Defense Acquisition Board/Major Automated Information System Review Committee/Council (DAB/ MAISRC) Review.

5.2 Overview

A general overview of the REA process is illustrated in Figure 5-1. Ground rules and assumptions are identified and provide management flexibility. A common Cost Element Structure (CES) is developed that captures all the costs associated with all the strategies. Estimating methods are identified, necessary data is collected, and estimates for each cost element are developed. The estimates are aggregated, converted into their Present Value, and used to calculate the NPV and BIR for each candidate strategy. Other economic indicators, such as Break-even Point (BP) and Rate of Return (ROR) can be used, but generally require additional effort. Sensitivity and risk analyses may be performed to identify the sensitivity and risk associated with the estimates.

The estimates of specific cost elements for each strategy should be based on the best information available. Vendor quotes or catalog prices are typically better sources than parametric estimating models or engineering judgment. However, when a parametric software estimating model is used to estimate the costs of specific cost elements, it is imperative to use the same model for the estimates of the same cost element for all the other strategies.

The estimate for each strategy should be made at the appropriate level. This can be at the system level, the Computer Software Configuration Item (CSCI) level for embedded systems, or the computer program level for MIS systems. The CSCI or program level is generally the most appropriate level. Estimates at the system level should be avoided if possible because they may be at too gross a level and thus inaccurate. Estimates are not generally made below the CSCI level (at the computer software unit (CSU) level or computer software component (CSC) level), unless this is where the specific need is, because they may be at too detailed a level and require unnecessary effort.

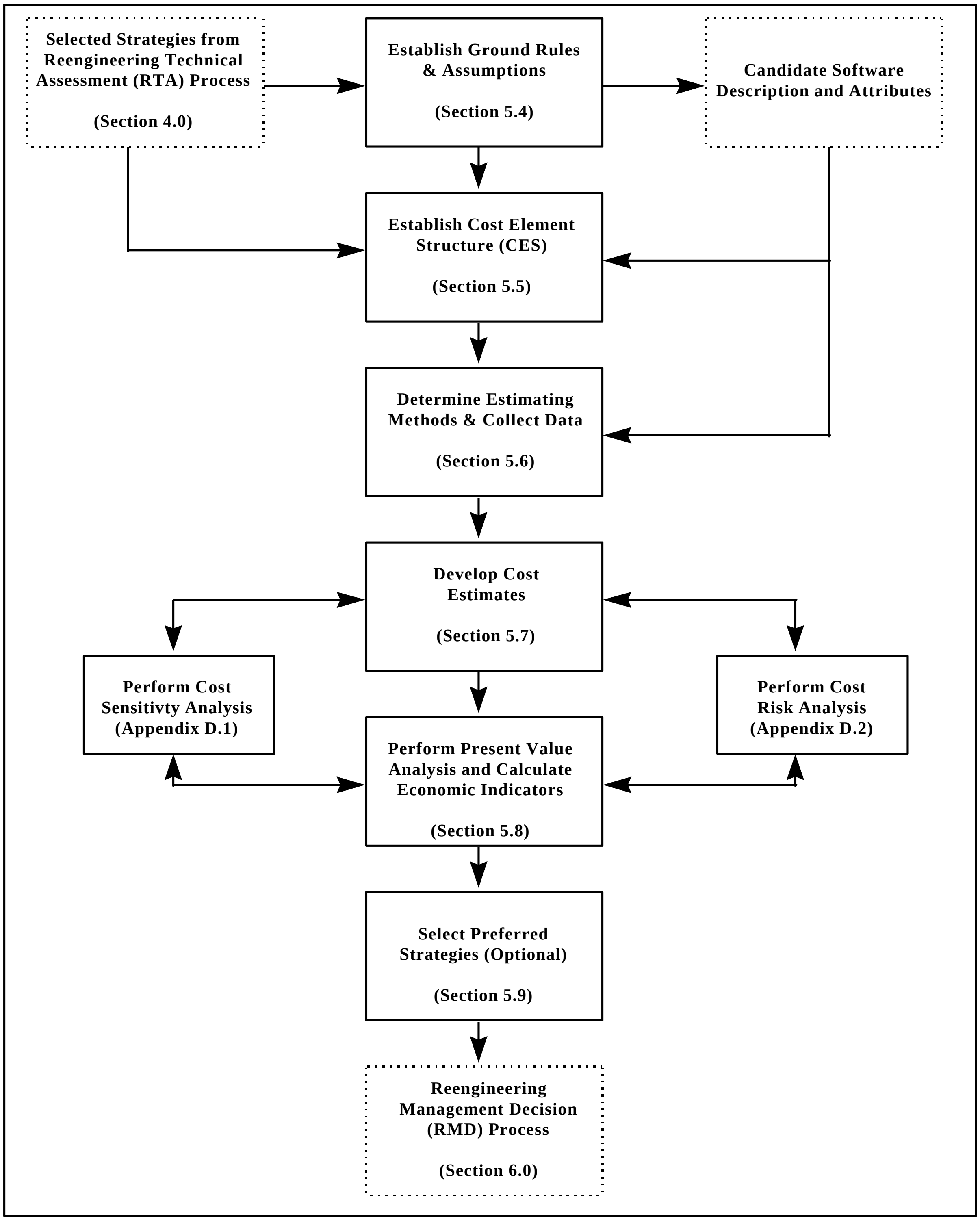

Selected Strategies from Reengineering Technical Assessment (RTA) Process
(Section 4.0)
Establish Ground Rules & Assumptions
(Section 5.4)
Candidate Software Description and Attributes
Establish Cost Element Structure (CES)
(Section 5.5)
Determine Estimating Methods & Collect Data
(Section 5.6)
Develop Cost Estimates
(Section 5.7)
Perform Cost Sensitivty Analysis (Appendix D.1)
Perform Cost Risk Analysis (Appendix D.2)
Perform Present Value Analysis and Calculate Economic Indicators
(Section 5.8)
Select Preferred Strategies (Optional)
(Section 5.9)
Reengineering Management Decision (RMD) Process
(Section 6.0)

Figure 5-1. Reengineering Economic Assessment (REA) Process Overview

5.3 Step-by-Step Approach

This section provides a simplified description of each of the major steps required to accomplish an REA. The reader is referred to sections 5.4 through 5.9 for supporting details and theory. The objective of the economic assessment is to prepare a cost estimate for each cost element by fiscal year using constant dollars. The output of the REA is a report that includes the economic information required for the Detailed Assessment Results (DAR) worksheet as shown in Table C-1.

Step 1: Establish Ground Rules & Assumptions (Refer to Section 5.4): Ground Rules and Assumptions (GR&As) are established and documented to provide a consistent basis for all estimates. Figure B-1 (found in Appendix B) is an example of a GR&As Worksheet that can be tailored for use on a specific project.

Step 2: Establish Cost Element Structure (Refer to Section 5.5): A common Cost Element Structure (CES) is established for estimating all candidates and strategies. A CES is similar to a work breakdown structure. Figure 5-2 is an example of a generic CES. The CES should include cost elements that capture all costs associated with acquiring, developing, and/or maintaining all strategy-specific software, hardware, software engineering tools, training, and other elements. Only costs incurred from the beginning of the reengineering effort to the end of the software's useful life are included. Sunk costs are not included. Investment costs are usually not required for the Status Quo (Strategy 1). Costs that are identical for all strategies (referred to as wash costs) must be included for all strategies. Costs for Operations & Support (O&S) of the Status Quo during the reengineering period (referred to as phase-out costs) must be included in the O&S costs for all reengineering strategies.

Step 3: Determine Estimating Methods & Collect Data (Refer to Section 5.6): Estimating methods are determined for each cost element. Methods that are typically used include level of effort, catalog prices or written quotations, engineering estimates (bottom-up), expert opinion, analogy, and/or parametric models. Once an acceptable estimating method is identified for each CES cost element, the analyst must collect the data necessary to actually perform the estimating calculations. The analyst should strive to obtain the most accurate, precise, current, and applicable prices available. However, good judgment should be used in determining the amount of effort expended to obtain the best prices. In order to develop an REA in a timely manner, engineering estimates can be used in place of prices that are difficult and time consuming to obtain.

Step 4: Develop Cost Estimates (Refer to Section 5.7): The worksheets shown in Appendix B can be used to consolidate the various cost element estimates. Once the annual costs in constant-dollars for all CES elements have been determined for each strategy, the results can be aggregated to the appropriate higher level (CES 1.0, 2.0 or 3.0) annual totals. Simple multiplication by the appropriate discount factors will convert each annual total to its present value. When using parametric models, the analyst should use the same model for all strategies and candidate software that will be compared. Parametric models may need to be adjusted for reengineering versus new development.

Step 4.1: (Optional): Perform Cost Sensitivity Analysis (Refer to Appendix D.1): The analyst may perform cost sensitivity analyses on the primary variables used by parametric estimating models to measure how sensitive the estimate is to changes in the variable. This typically involves iteratively applying the model while varying the value of the parameter being evaluated.

Step 4.2: (Optional): Perform Cost Risk Analysis (Refer to Appendix D.2): The analyst may perform cost risk analysis to adjust the estimates for each strategy to the same confidence level.

Step 5: Perform Present Value Analysis and Calculate Economic Indicators (Refer to Section 5.8): The analyst calculates the NPV and BIR of each strategy. This is done by using the definitions provided in Section 5.8. Other economic indicators may be used to determine the most cost effective strategy including Break-even Point (BP) and Rate of Return (ROR). The present value of total cost and present value of total benefit may also be considered. The bottom line numbers from the cost worksheets are entered into the appropriate cells in the DAR worksheet (Table C-1). The REA analyst meets with the RTA team to co-author the Management Report as described in Section 6.

Step 6: (Optional) Select Preferred Strategies (Refer to Section 5.9): The analyst may economically rank-order the proposed strategies using an appropriate economic indicator prior to meeting with the technical personnel for the reengineering management decision process.

5.4 Establish Ground Rules and Assumptions (GR&As)

All estimates are based on the same Ground Rules and Assumptions (GR&As). A worksheet containing examples of typical ground rules and assumptions is provided in Figure B-1. These should be tailored to each specific project. GR&As provide part of the framework upon which the estimate is developed. They describe how technical issues relate to the estimating process and state how specific issues will be treated. They also provide a method for addressing uncertainty by permitting the analyst to assume a response to a required input or factor when the actual value or response is unknown. These must be specifically stated so that the decision maker understands the basis for the estimate. For example, some parametric software estimating models may include a variable for which the analyst does not have an input. Frequently this can be resolved by making an assumption based on professional judgment and a thorough understanding of the program being estimated. Also, your organization may have a set of organizational default values or criteria for choosing these values.

5.5 Establish Cost Element Structure (CES)

The REA for each candidate and strategy being considered must be based on a common Cost Element Structure (CES) for all candidates and strategies. A generic CES for software is provided in Figure 5-2 and in Appendix B based on *MIL-HDBK-171, Work Breakdown Structure for Software Elements*. A detailed CES applicable to a Major Automated Information System (MAIS) is provided in the *USA Economic Analysis Manual*, Appendix D. Investment (CES 1.0) is defined as the effort required to reengineer the software. Investment costs are usually not required for Status Quo. If any, they should be included in O&S for Status Quo (CES 3.0). O&S includes operations, support (maintenance), training, and other elements for the remaining life of the software. The CES for O&S is essentially the same as for Investment since the process is similar, but on a smaller scale. The O&S cost for reengineered software includes support of the reengineered software (CES 2.0) as well as continued support (phase-out) of the Status Quo (CES 3.0) during the reengineering period.

- 1.0 Investment for Reengineered Software
  - 1.1 Software Development
    - 1.1.1 Requirements Analysis
    - 1.1.2 Preliminary Design
    - 1.1.3 Detailed Design
    - 1.1.4 Code & Unit Test
    - 1.1.5 Unit Integration & Test
    - 1.1.6 CSCI Test
    - 1.1.7 SPCR Resolution
    - 1.1.8 Other
  - 1.2 CSCI-CSCI Integration & Test
  - 1.3 System Integration & Test
  - 1.4 Training
  - 1.5 Data
  - 1.6 Peculiar Support Equipment
  - 1.7 Operational Site Activation
  - 1.8 Facilities & Utilities
  - 1.9 Hardware
  - 1.10 System Operations
  - 1.11 Independent Verification & Validation (IV&V)
  - 1.12 System Engineering/Program Management
  - 1.13 Other
- 2.0 Operations & Support for Reengineered Software
  - (Repeat same sub-elements as 1.0)
  - 2.14 O&S for Status Quo during Reengineering
- 3.0 Operations & Support for Status Quo
  - (Repeat same sub-elements as 1.0)

Figure 5-2. Generic Cost Element Structure (CES) for Software Reengineering

Using a common CES will ensure that all costs for each strategy are considered in a complete and consistent manner. The CES should include cost elements that capture all costs associated with acquiring, developing, and/or maintaining all strategy-specific software, hardware, software engineering tools, training, and other elements. Hardware costs can drive the need to reengineer even though the software maintenance costs are low. Systems with high or potentially high hardware maintenance cost may need to be reengineered because the hardware is too expensive to maintain.

Costs that are identical for all strategies (referred to as wash costs), must still be included for each strategy. The reason for including "wash costs" is that the cost estimates developed for the decision maker are usually translated into current year, budgetary documents and should reflect all elements which require funding. Sunk costs (past expenditures or irrevocably committed costs which are unavoidable) are not included. Costs for O&S of the Status Quo during the reengineering project (typically known as phase-out costs) must be included in the O&S costs for all reengineering strategies.

In addition to tailoring the CES for the specific costs to be estimated, the CES may be tailored to satisfy any of the following requirements:

- To match the CES from a useful and earlier estimate for this program,
- To correlate the CES output from the cost estimating model used, or
- To highlight a particular cost sensitivity, e.g., to provide more detailed information.

Further guidance on development of a CES can be found in the references identified in Section 2.0, paragraphs 2.1.1 a–c and 2.1.2 b, c, and f.

5.6 Determine Estimating Methods and Collect Data

Complete software development and support estimates typically use a variety of estimating methods including level of effort, catalog prices or quotations, engineering (bottom-up), expert opinion, analogy, and/or parametric models. Different estimating methods are typically used for different CES elements. For example, when the specific items are relatively well known or defined, vendor contract prices or quotations can frequently be used to estimate the cost of purchased training (CES 1.4), software development tools (CES 1.6), and hardware (CES 1.9). Analogies can be used when the specific items are unknown, but the relative costs for similar items are known from a similar prior effort.

Software development effort (CES 1.1) and related efforts (CES 1.2, 1.3, etc.) are typically estimated using parametric models, but may also be estimated based on analogy (in-house historical data) or best engineering estimates. The analyst doing the REA has several parametric models from which to choose. These include but are not limited to: CHECKPOINT, PRICE-S, SEER-SEM, SLIM, and SOFTCOST-OO. SASET was determined to be not applicable to reengineering.

All of these models are flexible enough to estimate the range of strategies (Status Quo and various reengineering strategies) as described in Appendices E through J. However, these models do not include all the sub-elements of CES 1.0 in their estimates. Depending on the model used, it may be necessary to estimate some of the sub-elements of CES 1.0 separately. There is no single "best" software development estimating model. The analyst is encouraged to select a model based on his/her familiarity and skill with the model, the availability of the model, the availability of input data, and the suitability of the model CES. Desired characteristics of a model include: adaptability to reengineering, ease of use, adaptability to an organization's actual historical data, accurate estimates of cost and schedule, and applicability to maintenance estimation.

Software O&S effort (CES 2.0 and 3.0) can be estimated either by using a parametric model or by performing an engineering estimate of the personnel and resources required. For existing software, the future software support effort can also be estimated by extending the actual historic support costs into the future using either a straight-line projection or by adjusting the individual annual cost estimates based on perceived changes in the support requirements.

The remaining life (YL) for all candidate strategies is the same as that established by the software operational requirements. This includes the Status Quo. Remaining life is defined as the number of years required for the Investment effort (YI) plus the number of years for the O&S effort (YS). YI is typically estimated by the model and YS is the difference between YL and YI.

$$YL = YI + YS \qquad or \qquad YS = YL - YI$$

Once an acceptable estimating method is identified for each CES cost element, the analyst must collect the data necessary to actually perform the estimating calculations. For the parametric models this includes gathering information for each of the input variables for the model. The specific values that will be used as inputs to a parametric estimating model are typically identified both from the information required to perform the RTA and from other sources familiar with or responsible for the candidate software. For other CES elements, this represents collecting vendor quotations, seeking pricing information from catalogs, and working with engineers to develop "bottom-up" engineering estimates. The analyst must have either a thorough understanding of the system or access to someone who has this knowledge, and be able to validate the data collected as to its accuracy, volatility, or appropriateness.

5.7 Develop Cost Estimates

The worksheets shown in Appendix B can be used to consolidate the various cost element estimates and economic indicators. Once the annual costs in constant dollars for all CES elements have been determined for each strategy, the results can be aggregated to the appropriate higher level (CES 1.0, 2.0, or 3.0) annual totals. Simple multiplication by the appropriate discount factor will convert each annual total to its present value. If desired, the annual costs in constant dollars can also be multiplied by the inflation rate to convert each annual total to its current (then year) value used for break-even analysis and budgeting. Additional worksheets may be prepared to vary the discount rate or inflation rate for variance analyses. The group of worksheets shown in Appendix B should be completed for each reengineering strategy being considered for a program.

Analysts might want to perform cost sensitivity analyses on the primary variables used by parametric estimating models to measure how sensitive the estimate is to changes in the variable. This typically involves repeatedly applying the model while varying the value of the parameter being evaluated. A sensitivity analysis assists the analyst and decision maker in assessing changes based on programmatic factors. For example, if the remaining life potentially could vary between 8 and 10 years, then the analyst could repeat the estimate using both the 8 and 10 year remaining life assumptions to determine if this change would affect the results of the analysis while holding the other parameters constant. Appendix D, Table D-1, provides an example of performing cost sensitivity analysis. Further information on sensitivity analysis can be found in the *USA Economic Analysis Manual*.

In addition, analysts might want to perform a cost risk analysis to adjust the estimates for each strategy to the same confidence level. Risks can be managed by estimating a dollar value for each risk whether the risk is technical, schedule, or cost. *DoD's 5000.2-R* offers seven factors for analyzing risks: threat, technology, design, support, manufacturing, cost, and schedule. The purpose of risk analysis is twofold: (1) to determine the likelihood that the risk event will take place, and (2) to determine the impact of the risk event. That is, the analysts need to measure the probability that the event will occur and also measure what the impact of the event will be. A cost risk analysis uses standard risk analysis techniques but uses costs as the measure of the "impact". Although this handbook is not intended to provide a study in risk assessment, Appendix D, Table D-2, provides an example of performing cost risk analysis. Further information on cost risk analysis can be found in the *AFSC Cost Estimating Handbook* and in the *USA Economic Analysis Manual*. Information on general risk management techniques can be found in *Software Engineering, Risk Analysis and Management* by Robert Charette and AFMC Pamphlet 800-45, *Software Risk Management*.

5.8 Perform Present Value Analysis and Calculate Economic Indicators

Since the purpose of the REA is to compare the relative future costs and benefits of each strategy, Present Value Analysis is used. This process equitably compares the value of different annual future costs and benefits in common terms. The method for calculating present value is described in Appendix B. The discount rate is dependent upon the period of analysis and employs the real (net inflation) discount rates (as opposed to the nominal discount rates which include inflation) for discounting constant dollar flows of money. The current discount rate is provided in Appendix C of OMB Circular A-94 which is available upon request from the office of Economic Policy in OMB at (202) 395-3381 or at http://www.whitehouse.gov/wh/eop/omb. An example of present value analysis and the use of economic indicators is provided in the *USA Economic Analysis Manual*, Appendix I.

The REA uses the Total Cost (TC) and Total Benefit (TB) for comparing the costs and benefits for each strategy. Total Cost includes only costs that are incurred from the beginning of the reengineering effort to the end of the software s useful life. It specifically excludes all costs incurred prior to the beginning of the effort (i.e., sunk costs). In contrast, a Life Cycle Cost Estimate (LCCE) would include the sunk costs. Excluding sunk costs, the general estimating process for developing Total Cost estimates is very similar to developing an LCCE.

The Present Value of the Total Cost (PV(TC)), and its derivative measures are used in the REA to compare the costs of alternative strategies. Similarly, the Present Value of the Total Benefit (PV (TB)) is used to compare the benefits of alternative strategies. Since Present Value is used rather than inflation adjustments, etc., the Present Value estimates prepared in the REA process cannot be used to support development of a budget or for DAB/MAISRC milestone reviews. Budgets can be prepared by using constant dollars and applying prescribed escalation factors.

The REA of candidate strategies depends on estimating and comparing the PV(TC) and PV(TB) of each strategy using a variety of economic indicators. The preferred economic indicators are NPV and BIR. PV(TC) and PV(TB) also can be used. Other economic indicators, such as BP and ROR can be used, but generally require additional effort. The economic indicators are calculated for each strategy and recorded on the DAR worksheet per Section 6.

Present Value of Total Cost (PV(TC)): The cumulative total of Investment (CES 1.0) and Operations & Support (CES 2.0 or 3.0) costs stated in terms of Present Value. It includes all costs incurred from the beginning of the reengineering effort through the end of the software s life, excluding sunk costs. For all strategies other than Status Quo (Strategy 1), Total O&S cost includes O&S of Strategy N after reengineering, O&S_N, and O&S (phase-out) of Status Quo during the reengineering period, O&S_{1DR}.

$$PV(TC_N) = PV(Total\ Investment_N) + PV(Total\ Operations\ \&\ Support_N)$$

$$= PV(TI_N) + PV(TO\&S_N)$$

where N is the number of the strategy.

Present Value of Total Benefit (PV(TB)): The difference between the Present Value of the Total O&S cost of Status Quo and the Present Value of the Total O&S cost of Strategy N. For all strategies other than Status Quo, Total O&S cost includes O&S of Strategy N after reengineering, O&S_N, and O&S (phase-out) of Status Quo during the reengineering period, O&S_{1DR}.

$$PV(TB_N) = PV(TO\&S_1) – PV(TO\&S_N)$$

Net Present Value (NPV): Within the context of the above two defined economic indicators (total cost and total benefit), NPV is the difference between the Present Value of the Total Benefit (PV(TB)) and the Present Value of the Total Investment (PV(TI)). It represents the net difference between dollars saved or avoided and dollars invested over time expressed in equivalent amounts at a common point in time.

$$NPV_N = PV(TB_N) – PV(TI_N)$$

This equation for NPV may also be expressed as:

$$NPV_N = PV(TC_1) – PV(TC_N)$$

The preferred strategy is the strategy with the highest NPV. (*USA Economic Analysis Manual*, paragraphs 5-1, 5-2a(3), and 5-3c)

Benefit Investment Ratio (BIR): The ratio of benefit to investment. This definition of BIR is consistent with DoD definitions wherein investment measures the cost of software development and related efforts (CES 1.0) and investment does not include the O&S cost for either Status Quo or Strategy N. (*USA Economic Analysis Manual*, paragraph 5-3d)

$$BIR_N = \frac{PV(TB_N)}{PV(TI_N)}$$

Break-even Point (BP): The break-even point where the total cost of one strategy is equal to the total cost of the other. Also, where the Benefit is equal to the Investment. (*USA Economic Analysis Manual*, paragraph 5-3b)

BP = that point in time at which:

$$TC_1 = TC_N \text{ or } TB_N = TI_N$$

in current (inflated) dollars

Rate of Return (ROR): The interest rate at which the Present Value of the Total Benefit is equal to the Present Value of the Total Investment (i.e., NPV = 0 and BIR = 1.0) through the remaining life of the strategy being evaluated. (*USA Economic Analysis Manual*, paragraph 5-3f)

$$ROR = i \text{ at which } PV(TI_N) = PV(TB_N) \text{ or}$$

$$\frac{\text{Investment (year 1)}}{(1+i)} + \frac{\text{Investment (year 2)}}{(1+i)^2} + \dots = \frac{\text{Benefit (year 1)}}{(1+i)^1} + \frac{\text{Benefit (year 2)}}{(1+i)^2} + \dots$$

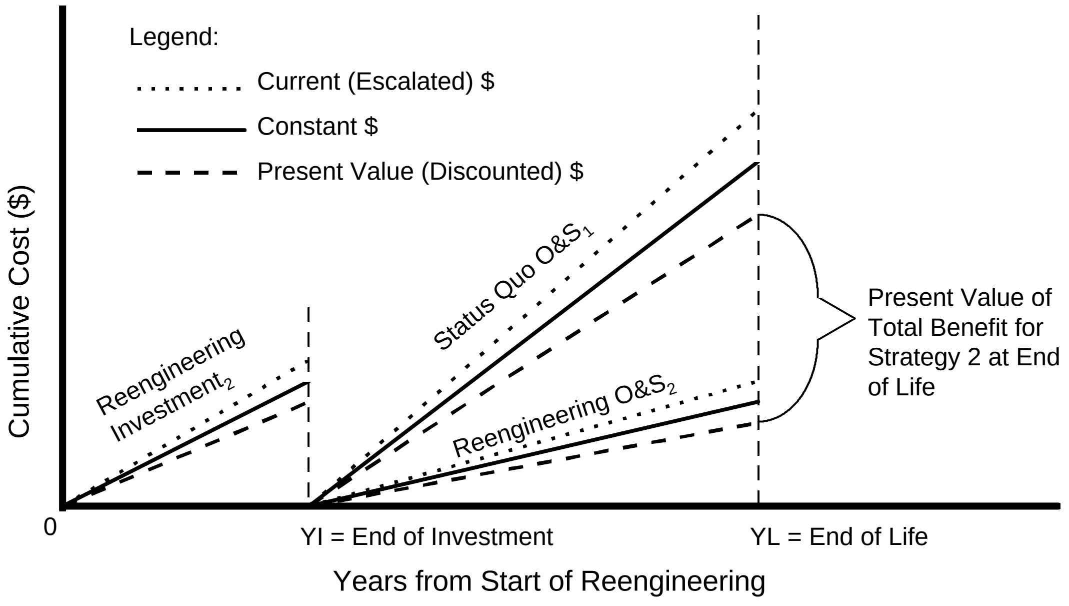
The above economic indicators, except BP, assume that all strategies have identical remaining economic lives. Where strategies have differing remaining lives, determine whether the longest or shortest life or some other time period is to be used as a basis for comparison, and make an adjustment for unequal life. If the shortest life is used to establish the time period of the analysis, recognize the residual values of the strategies with the longer lives in the cost computation. If the longest life is used, recognize the cost of extending the benefit-producing years of those strategies with a shorter life. Ensure that the decision maker is presented the complete and valid costs for each strategy for the entire length of the analysis. In cases where adjusting the remaining life is totally impractical, strategies with unequal lives may be compared based on equivalent (uniform) annual cost. (See Appendix B and the *USA Economic Analysis Manual*, paragraph 5-4a.)

Note that in the above equations for PV(TB) and NPV, the O&S cost for Status Quo during the reengineering period, $O\&S_{1DR}$ cancels out because it is present in the O&S cost of Status Quo and of Strategy N. Thus, the benefit begins accruing at the end of the reengineering investment period (YI), at the start of O&S of the reengineered software. The benefit is equal to the difference in O&S costs of Status Quo and Strategy N accruing from this point in time onward. This is graphically illustrated in Figure 5-3 for two hypothetical Strategies 1 and 2. NPV = 0 and BIR = 1 when the benefit equals the investment (in discounted $). Also, the break-even point occurs when the benefit equals the investment (in current $). For a strategy to be economically viable, these points must occur prior to the end of remaining life.

5.9 Select Preferred Strategies (Optional)

From a purely economic perspective, NPV is typically considered the most appropriate indicator of the "best" strategy. (*USA Economic Analysis Manual*, paragraph 5-1, 5-2a(3), and 5-3c.) It reflects the strategy that results in the largest "net benefit," and is useful when the actual size of the return from the alternative is the concern. However, the NPV method of comparison normally favors larger size strategies.

The BIR is also considered an appropriate indicator to rank order the candidate strategies (*USA Economic Analysis Manual*, paragraph 5-1 and 5-2). In contrast to NPV, BIR emphasizes the relationship of the *ratio* of the benefit to the investment. It reflects the strategy that results in the largest benefit relative to the investment made to achieve the benefit. Caution is recommended when using BIR because it can lead to invalid results if the BIR is higher but the benefits are either lower or the investment is higher than another alternative. The magnitude of the benefit and investment is obscured by the ratio.



NPV = PV (Benefit) - PV(Investment)	BIR = PV(Benefit) / PV(Investment)
= PV(O&S$_1$) - PV (O&S$_2$) - PV(Investment$_2$)	= (PV(O&S$_1$) - PV(O&S$_2$)) / PV (Investment$_2$)
> 0: Positive Return (Discounted $)	> 1: Positive Return (Discounted $)
< 0: Negative Return (Discounted $)	< 1: Negative Return (Discounted $)
= 0: Break-even (Current $)	= 1: Break-even (Current $)

Figure 5-3. Net Present Value (NPV), Benefit Investment Ratio (BIR), and Break-even Point (BP)

In contrast to the NPV and BIR, ROR is independent of the discount rate. A simple minimum test for ROR is that it be greater than the interest rate (i.e., cost of capital for a corporation or the interest rate on Treasury Bills for the Government). The BP is useful as a tie-breaker when NPVs, BIRs, or RORs are nearly equal between alternatives. The earlier the BP the better. BP is not sensitive to the inaccuracies that may occur after the BP. Finally, the Total Cost or Benefit could also be used to rank order the candidate strategies. However, they only consider the gross cost or benefit rather than the net benefit (NPV) of the strategy.

6. Reengineering Management Decision Process

6.1 Introduction and Overview

The Reengineering Management Decision (RMD) process (Figure 6-1) is the final step in the Software Reengineering Assessment (SRA) process. This section of the SRAH will address how the Reengineering Technical Assessment (RTA) and Reengineering Economic Assessment (REA) teams should present their findings to management and suggest criteria for management to prioritize and authorize software reengineering projects. The RMD process consists of the following 3 steps:

- **Step 1: Prepare Management Report (refer to Section 6.2):** The Management Report is necessary because the reengineering project decision maker is usually not one of the persons performing the RTA or REA. The data from these sections needs to be formally presented to management in a consistent and professional format. Even if the reengineering project decision maker has a significant part in the execution of the previous two processes, this report should still be generated as a record of the SRAH results for the given candidate reengineering project. This report should be written by the members of the RTA and REA teams as a joint effort, and then used to select the reengineering project(s).

- **Step 2: Reengineering Projects Selection (refer to Section 6.3):** Selecting the reengineering project(s) consists of the reengineering project and resource allocation decisions based on assessment results and organizational factors. This selection is primarily intended for management personnel.

- **Step 3: Implement and Document Decision (refer to Section 6.4):** This section suggests the kinds of documentation that should accompany the final decision and implementation of any SRAH-recommended reengineering projects. This documentation and implementation is primarily intended for both management personnel and the reengineering project team.

6.2 Prepare Management Report

The Management Report consists of the following sub-sections:

- Executive Overview
- Body of the Report

- Appendices including:
  - Reengineering Technical Assessment (RTA) worksheet(s) (Results of Section 4)
  - Reengineering Economic Assessment (REA) worksheet(s) (Results of Section 5)
  - Detailed Assessment Results (DAR) worksheet(s)
  - Recommended Strategy Rank (RSR) worksheet(s)

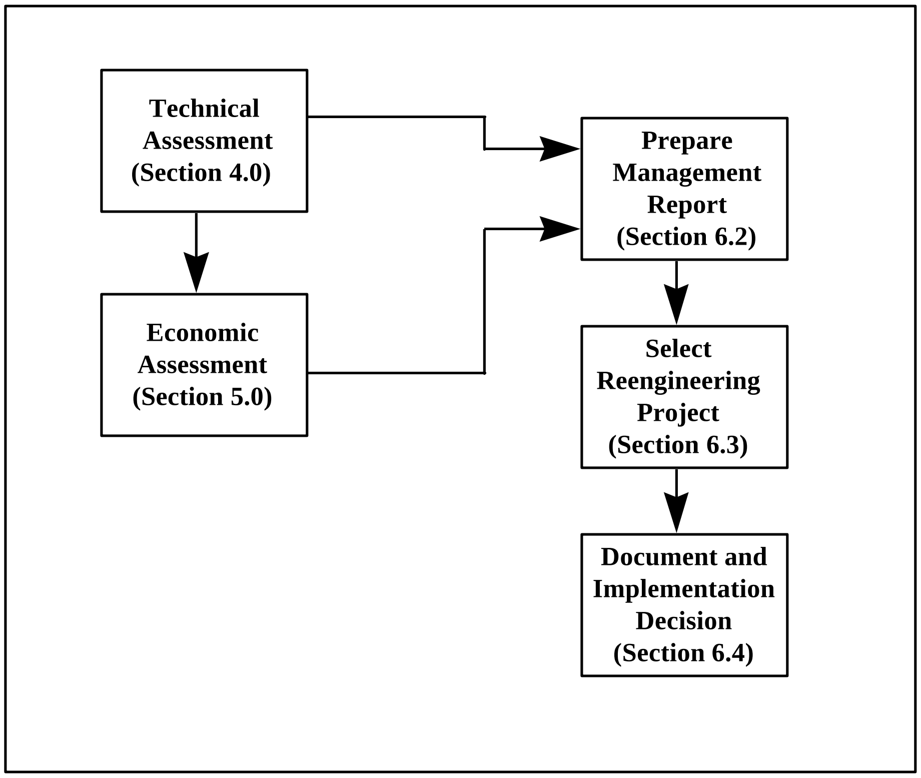

Technical Assessment (Section 4.0)
Economic Assessment (Section 5.0)
Prepare Management Report (Section 6.2)
Select Reengineering Project (Section 6.3)
Document and Implementation Decision (Section 6.4)

Figure 6-1. Reengineering Management Decision Process

Following completion of the RTA and REA worksheets, transfer the results to the DAR provided in Appendix C, then complete the RSR worksheet and the Executive Overview.

If another report format is desired, (i.e. if an organization has a standard managerial report format), then that alternative format can be used. The key issues are consistency and efficiency. All reports should look the same (for accurate comparison of results) and be in a format that the manager is comfortable with. The format provided in Appendix C is presented merely as a recommendation.

Detailed Assessment Results (DAR) Worksheet

Candidate Information | **Strategy Information** | **Candidate Composite Reengineering Strategy Total (CCRS)**

Candidate Software Name and Description

Special Considerations:

Strategy	Redocument	Restructure	Translate	Retarget	Data Reeng.	Reverse Eng.	Redevelop	Status Quo	Rank Within Candidate
RTA Stategy Average						Yes No	Yes No	Yes No	
Invest. Cost *								$0	
O&S Cost *								$0	
Total Cost *								$0	
Total Benefit (or ROI)								$0	
Rate of Return								N/A	
Break Even Point								N/A	
BIR								N/A	
NPV								$0	

Remain. Life	Cand. Age	Cand. Import.	Current Maint. Environ.	Reeng. Prep.

Recommended Ranking									

*Use Uniform Annual Cost when comparing strategiess or candidates of unequal life.

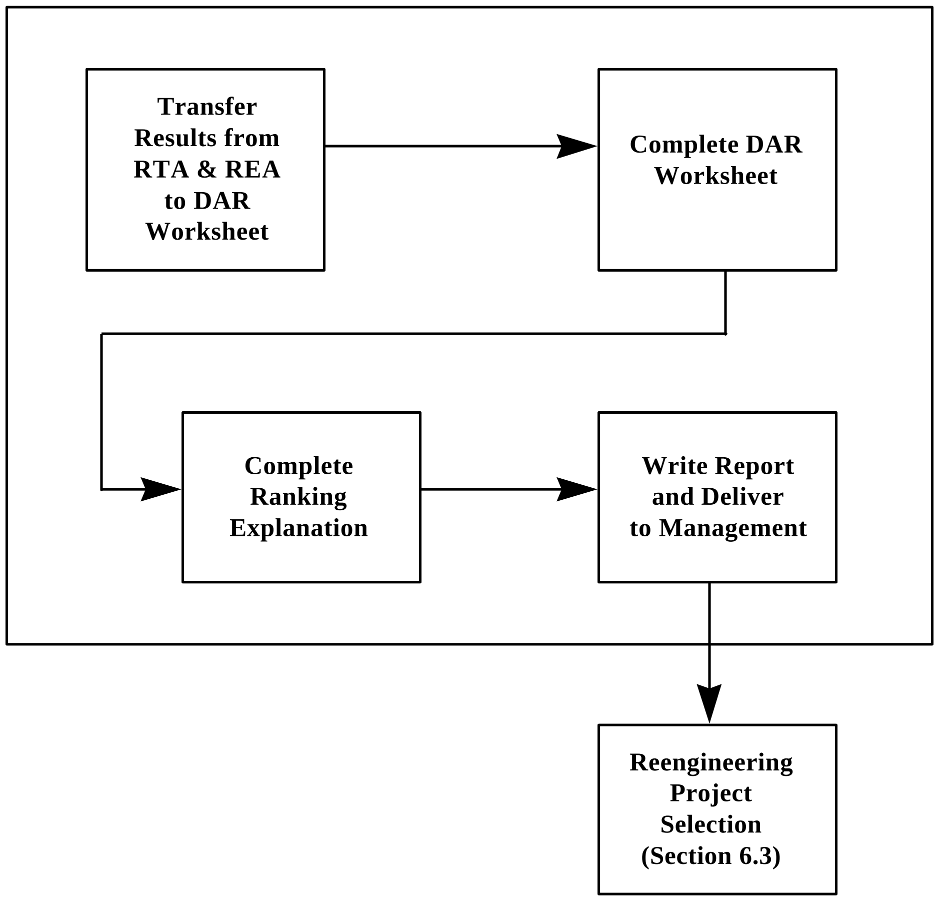

Transfer Results from RTA & REA to DAR Worksheet
Complete DAR Worksheet
Complete Ranking Explanation
Write Report and Deliver to Management
Reengineering Project Selection (Section 6.3)

Figure 6-2. Management Report Preparation Process

6.2.1 Detailed Assessment Results (DAR) Worksheet

After completing SRAH Sections 4 and 5 (RTA and REA worksheets), the reengineering technical and economic teams should meet, with completed worksheets, to finish the 4 sections of the DAR worksheet, provided in Appendix C.

- Candidate Information
- Strategy Information
- Recommended Ranking
- Candidate Composite Reengineering Strategy (CCRS) Total

6.2.1.1 Candidate Information

Fill out the Candidate Information section of the DAR worksheet using general knowledge about the candidate software and data from the RTA worksheets. Include any special considerations that management may need to be aware of to make their reengineering decision. For example, this might include references to organizational restructuring that may affect the candidate software. Only include information that might directly affect management's reengineering decision regarding this candidate software.

Data concerning remaining life, candidate age, candidate importance, current maintenance environment, and reengineering preparation come directly from the RTA worksheets.

6.2.1.2 Strategy Information

Transfer to the DAR worksheet the economic information from the REA and RTA worksheets for each candidate software and its associated strategies. The columns and their respective units are:

- RTA Strategy Average - 3 being the highest need for this strategy, and 1 being no need (from the RTA worksheet)
- Investment Cost - dollars
- Operations and Support (O & S) Costs - dollars
- Total Cost - dollars
- Total Benefit (or ROI) - dollars
- Rate of Return - ratio
- Break-even Point - years
- Benefit Investment Ratio (BIR) - no units, but the greater the ratio, the more desirable the strategy
- Net Present Value (NPV) - dollars

6.2.1.3 Recommended Ranking

Based on the Strategy Information and Candidate Information, Rank Within Candidate prioritizes the reengineering strategies for a given software candidate. Ranking varies from 1 (high priority) to 8 (low priority). Three methods for determining this ranking are:

- **Method 1: Committee Decision**

  The most successful method for ranking strategies and candidates is to have both teams jointly rank the strategies at their meeting. Ranking should be based on the technical and economic data (particularly the RTA Strategy Average, Net Present Value and Benefit Investment Ratio), and organizational data (see Section 6.3 for list of additional considerations). This method is recommended because it has been found in SRAH field tests that the technical and economic teams have the best understanding of the candidates and thus the best perception on strategy ranking.

- **Method 2: US Army Nonquatitative Benefits Process**

  If a completely quantitative ranking method is desired (not recommended due to significantly increased DAR worksheet complexity with minimal additional benefits or increased decision accuracy), Table C-3 (SRAH page C-4 and Department of the Army, *Economic Analysis Manual, July 1995*) provides a method to display and compare nonquatitative benefits, (e.g., data items that are not ordinarily quantitative by nature).

- **Method 3: Existing Organizational Process**

  Use an existing methodology within the organization for ranking software maintenance requests. This method should be documented in the Executive Overview of the Management Report.

6.2.1.4 Candidate Composite Reengineering Strategy (CCRS) Total

The CCRS Total helps to compare and contrast the sum total of all reengineering strategies for a given software candidate against other software candidates. Thus, even though a single reengineering strategy may not be sufficient to warrant reengineering, a combination of reengineering strategies may favor reengineering a candidate over a candidate with a single reengineering strategy indicated.

Begin by eliminating reengineering strategies whose data should not be used in calculating the CCRS Total. A strategy should be eliminated if:

- RTA Strategy Average is less than 2.00 or “no” is indicated
- NPV is negative
- BIR is less than 1

The remaining strategy values should be totaled and entered in the corresponding CCRS Total box. This is accomplished differently for each column depending on the method used to derive that column:

- RTA Strategy Average: sum the remaining strategy values
- Investment Cost, O&S Cost, Total Cost, and Total Benefit: sum the remaining strategy values
- Rate of Return sum and Break-even Point: average the remaining strategy values
- Benefit Investment Ratio (BIR): divide the CCRS Total Benefit by the CCRS Investment Cost
- NPV: add the NPV values for the remaining strategies

6.2.2 Recommended Strategy Rank (RSR) Worksheet

When multiple software candidates and their respective reengineering strategies are being evaluated, management requires a quick way to compare the myriad possible combinations and the resulting ranking recommendations from the technical and economic teams. The RSR worksheet in Appendix C provides, on a single sheet, the key information used to rank the candidate software and their associated reengineering strategies. The information required to complete the RSR worksheet can be found on the DAR worksheets.

Software candidates and their associated reengineering strategy should be ranked high if:

- The software is important (as determined by the Importance worksheet)
- The NPV is high (as determined by the economic analysis)
- The technical strategy is highly recommended (as determined by the technical analysis)

In addition, consideration should be given to a candidate's CCRS Total. Remember, the CCRS Total (i.e. the sum total of a candidate's reengineering strategies) may indicate a candidate's ranking is higher than if only one of its reengineering strategies is considered.

Recommended Strategy Rank (RSR) Worksheet

Candidate Software Name	Recommended Strategy	Strategy NPV	Candidate Importance	Ranking

6.2.3 Finishing the Management Report

Now that the DAR and RSR (provided in Appendix C) have both been completed, it's time to write the Management Report. This will include the items listed in Section 6.2: Executive Overview, Body of the Report, and Appendices containing the RTA worksheets, REA worksheets, DAR worksheets, and RSR worksheets.

6.2.3.1 Executive Overview

The Executive Overview should consist of a single page with the recommendations of the RTA and REA teams. Take the top RSR candidates and their reengineering strategies and put them into the Executive Overview. For each top candidate and strategy, explain the reasons why it is of prime consideration. These reasons should reflect the results of the RSR: importance, technical assessment, and NPV. Any additional key reasons for selecting these top candidates and strategies should also be included and justified.

6.2.3.2 Management Report Body

The body of the report should:

- Introduce and discuss the SRAH process used to formulate the Executive Overview findings
- Discuss an overview of the current maintenance environment
- Explain why software reengineering is being considered
- Describe the candidates analyzed
- Summarize the technical considerations for each reengineering strategy chosen
- Summarize the cost estimate findings

6.3 Reengineering Project Selection

In addition to the analysis work represented by the RSR, other criteria should be considered by management. Some of this criteria is discussed in the RTA but should be emphasized to management:

- Risk Analysis and organizational strategic goals
    - What do the users/customers think of the proposed reengineering effort?
    - If implemented, how will the reengineering project affect organizational budget, schedule, other projects, and organizational strategic goals? If not implemented, how will the candidate affect mission requirements or organizational strategic competitive advantage?

  - What effect will current mission requirements have on the reengineering effort (assuming the legacy software is in use while a copy of it is being reengineered)? Can the reengineering project be successful with constantly changing mission requirements (which would dictate changing the legacy software)?
  - How important is the candidate to the organization and will that candidate remain important long enough to recover the initial reengineering investment?
  - What are the external or political issues that affect the reengineering decision (and ultimately have an effect on the reengineering project's success)?

- Resource Analysis
  - Is funding available to support the reengineering effort?
  - Are internal personnel available to perform the reengineering effort? If not, can external personnel economically and satisfactorily complete the reengineering effort?
  - Are the personnel chosen for the reengineering effort (internal or external) adequately trained and knowledgeable of organizational and reengineering goals?
  - Do the tools exist to aid the reengineering effort? If so, what does the reengineering team think of these tools, and if not, can that part of the reengineering effort be accomplished cost effectively if done manually?
  - What does the proposed reengineering team think of the proposed projects? Reengineering involves change, and some people resist change, which could result in a less than enthusiastic approach to the reengineering project (which might cause the project to fail). Successful reengineering projects have reengineering teams that embrace change and are personally ***dedicated*** to completing the reengineering project successfully. Also, an understanding of the teams' opinion on the proposed project should have a direct impact on how the project is planned and implemented.

6.4 Implement and Document Decision

After the reengineering decisions have been made and projects are authorized or denied, then an addendum to the Management Report should be written describing in detail the reasons behind the decision to reengineer or not. As mentioned above, this is needed to understand the reasoning behind the current decision for facilitating future SRAH-based reengineering decisions.

If reengineering projects are initiated following an SRAH candidate analysis, then upon completion of those projects an additional Management Report addendum should be written describing the project results. This is necessary to calibrate the SRAH process to your organization based on correlating predictions to project results.

If the decision is made to authorize a reengineering project(s), Section 2.3 contains references to reengineering preparation and project planning models. These documents can help in the implementation of the reengineering project(s) and increase the probability of reengineering success and efficiency.

Finally, if both the pre-project and post-project addendums are written and added to the Management Report, the STSC would be very interested in this project data. Such data would help to calibrate and modify future versions of the SRAH. Contact the STSC at:

Software Technology Support Center
SRAH Reengineering Survey
OO-ALC/TISEC
7278 4th Street
Hill Air Force Base, Utah 84056

(801) 775-5555 x 3054 or DSN 775-5555 x 3054
re-eng@software.hill.af.mil